Sebastian Thrun
Michael Montemerlo
Stanford AI Lab
Stanford University
{thrun,mmde}@stanford.edu

The GraphSLAM Algorithm with Applications to Large-Scale Mapping of Urban Structures

Abstract

This article presents GraphSLAM, a unifying algorithm for the offline SLAM problem. GraphSLAM is closely related to a recent sequence of research papers on applying optimization techniques to SLAM problems. It transforms the SLAM posterior into a graphical network, representing the log-likelihood of the data. It then reduces this graph using variable elimination techniques, arriving at a lower-dimensional problems that is then solved using conventional optimization techniques. As a result, GraphSLAM can generate maps with 10^8 *or more features. The paper discusses a greedy algorithm for data association, and presents results for SLAM in urban environments with occasional GPS measurements.*

KEY WORDS—SLAM, robot navigation, localization, mapping

1. Introduction

In recent years, there have been a number of projects seeking to map physical environments with moving sensor platforms. Classical work includes mapping from the air (Konecny 2002), the ground (Elfes 1987; Moravec 1988), and underwater (Williams, Dissanayake, and Durrant-Whyte 2001). It includes indoor (El-Hakim et al. 1997; Iocchi, Konolige, and Bajracharya 2000), outdoor (Teller et al. 2001), and subterranean mapping (Baker et al. 2004). The development of techniques for the acquisition of such maps has been driven by a number of desires. They include photo-realistic rendering (Allen and Stamos 2000; Bajcsy, Kamberova, and Nocera 2000), surveillance (Wang, Thorpe, and Thrun 2003), scientific measurement (Baker et al. 2004), and robot guidance (Williams, Dissanayake, and Durrant-Whyte 2001). Not surprisingly, some of the primary work in this area has emerged from a number of different scientific fields, such as photogrammetry, computer vision (Tomasi and Kanade 1992; Pollefeys, Koch, and Gool 1998; Soatto and Brockett 1998), computer graphics (Levoy 1999; Rusinkiewicz and Levoy 2001), and robotics (Dissanayake et al. 2001).

In the SLAM community (SLAM is short for simultaneous localization and mapping), filter techniques such as the well-studied extended Kalman filter (EKF) have become a method of choice for model acquisition. The EKF was introduced mathematically by Cheeseman and Smith (1986), and implemented by Moutarlier and Chatila (1989a). This research has led to hundreds of extensions in recent years. Some of these approaches map the posterior into sparse graphical structures (Bosse et al. 2003; Paskin 2003; Thrun et al. 2002), to gain computational efficiency in the filtering process.

However, a key disadvantage of a filter technique is that data is processed and then discarded. This makes it impossible to revisit all data at the time of map building. Offline techniques, introduced by Lu and Milios (1997) and a number of follow-up papers (Golfarelli, Maio, and Rizzi 1998; Duckett, Marsland, and Shapiro 2000; Frese and Hirzinger 2001; Konolige 2004), offer improved performance by memorizing all data and postponing the mapping process until the end. Following observations in Golfarelli, Maio, and Rizzi (1998), the posterior of the *full SLAM problem* naturally forms a *sparse graph*. This graph leads to a sum of nonlinear quadratic constraints. Optimizing these constraints yields a maximum likelihood map and a corresponding set of robot poses.

This article represents a novel algorithm for mapping using sparse constraint graphs, called *GraphSLAM*. The basic intuition behind GraphSLAM is simple: GraphSLAM extracts from the data a set of soft constraints, represented by a sparse graph. It obtains the map and the robot path by resolving these constraints into a globally consistent estimate. The constraints are generally nonlinear, but in the process of resolving them they are linearized and the resulting least squares problem is

The International Journal of Robotics Research
Vol. 25, No. 5–6, May–June 2006, pp. 403-429
DOI: 10.1177/0278364906065387

Figures appear in color online: http://ijr.sagepub.com

solved using standard optimization techniques. We will describe GraphSLAM both as a technique for building a sparse graph of nonlinear constraints, and as a technique for populating a sparse "information" matrix of linear constraints.

When applied to large-scale mapping problems, we find that GraphSLAM can handle large number of features, and even incorporate GPS information into the mapping process. These findings are based on data acquired by a mobile robot system built to acquire 3-D maps of large-scale urban environments.

This article is organized as follows. We begin with an extended review of the literature. We then describe GraphSLAM intuitively, and characterize it both using graph-theoretical and information-theoretical terms. We state the basic algorithm and derive it mathematically from first principles. We then extend to address the data association problem. Finally, we present experimental results and discuss future extensions of this approach.

2. Related Work

In robotics, the SLAM problem was introduced through a seminal series of papers by Cheeseman and Smith (1986); Smith and Cheeseman (1986); Smith, Self, and Cheeseman (1990). These papers were the first to describe the well-known EKF SLAM algorithm, often used as a benchmark up to the present day. The first implementations of EKF SLAM were due to Moutarlier and Chatila (1989a,b) and Leonard and Durrant-Whyte (1991), some using artificial beacons as landmarks. Today, SLAM is a highly active field of research, as a recent workshop indicates (Leonard et al. 2002).

The first mention of relative, graph-like constraints in the SLAM literature goes back to Cheeseman and Smith (1986) and Durrant-Whyte (1988), but these approaches did not perform any global relaxation, or optimization. The algorithm presented in this paper is loosely based on a seminal paper by Lu and Milios (1997). They were historically the first to represent the SLAM prior as a set of links between robot poses, and to formulate a global optimization algorithm for generating a map from such constraints. Their original algorithm for globally consistent range scan alignment used the robot pose variables as the frame of reference, which differed from the standard EKF view in which poses were integrated out. Through analyzing odometry and laser range scans, their approach generated relative constraints between poses that can be viewed as the edges in GraphSLAM; however, they did not phrase their method using information representations. Lu and Milios's (1997) algorithm was first successfully implemented by Gutmann and Nebel (1997), who reported numerical instabilities, possibly due to the extensive use of matrix inversion. Golfarelli, Maio, and Rizzi (1998) were the first to establish the relation of SLAM problems and spring-mass models, and Duckett, Marsland, and Shapiro (2000, 2002) provided a first efficient technique for solving such problems. The relation between covariances and the information matrix is discussed in Frese and Hirzinger (2001). Araneda (2003) developed a more detailed elaborate graphical model.

The Lu and Milios algorithm initiated a development of offline SLAM algorithms that up to the present date runs largely parallel to the EKF work. Gutmann and Konolige combined their implementation with a Markov localization step for establishing correspondence when closing a loop in a cyclic environment. Bosse et al. (2003, 2004) developed Atlas, which is a hierarchical mapping framework based on the decoupled stochastic mapping paradigm, which retains relative information between submaps. It uses an optimization technique similar to the one in Duckett, Marsland, and Shapiro (2000) and GraphSLAM when aligning multiple submaps. Folkesson and Christensen (2004a,b) exploited the optimization perspective of SLAM by applying gradient descent to the log-likelihood version of the SLAM posterior. Their *Graphical SLAM* algorithm reduced the number of variables to the path variables—just like GraphSLAM—when closing the loop. This reduction (which is mathematically an approximation since the map is simply omitted) significantly accelerated gradient descent.

Konolige (2004) and Montemerlo and Thrun (2004) introduced *conjugate gradient* into the field of SLAM, which is known to be more efficient than gradient descent. Both also reduced the number of variables when closing large cycles, and report that maps with 10^8 features can be aligned in just a few seconds. Frese, Larsson, and Duckett (2005) analyzed the efficiency of SLAM in the information form, and developed highly efficient optimization techniques using multi-grid optimization techniques. They reported speedups of several orders of magnitude; the resulting optimization techniques are presently the state-of-the-art.

It should be mentioned that the intuition to maintain relative links between local entities is at the core of many of the submapping techniques discussed in the previous section—although it is rarely made explicit. Authors such as Guivant and Nebot (2001), Williams (2001), Bailey (2002) and Tardós et al. (2002) report data structures for minuting the relative displacement between submaps, which are easily mapped to information theoretic concepts. While many of these algorithms are filters, they nevertheless share a good amount of insight with the graphical information form discussed in this paper.

To our knowledge, the GraphSLAM algorithm presented here has never been published in the present form. However, GraphSLAM is closely tied to the literature reviewed above, building on Lu and Milios's (1997) seminal algorithm. The name *GraphSLAM* bears resemblance to the name *Graphical SLAM* by Folkesson and Christensen (2004a); we have chosen it for this paper because graphs of constraints are the essence of this entire line of SLAM research. A number of authors have developed *filters* in information form, which address the online SLAM problem instead of the full SLAM problem. These algorithms will be discussed in the coming paper, which explicitly addresses the problem of filtering.

Graph-like representations have also been applied in the context of SLAM filtering algorithms. In 1997, Csorba developed an information filter that maintained relative information between triplets of three landmarks. He was possibly the first to observe that such information links maintained global correlation information implicitly, paving the way from algorithms with quadratic to linear memory requirements. Newmann (2000) and Newman and Durrant-Whyte (2001) developed a similar information filter, but left open the question of how the landmark-landmark information links are actually acquired. Under the ambitious name '*consistent, convergent, and constant-time SLAM*,' Leonard and Newman further developed this approach into an efficient alignment algorithm, which was successfully applied to an autonomous underwater vehicle using synthetic aperture sonar (Newman and Rikoski 2003). Another seminal algorithm in the field is Paskin's (2003) *thin junction filter* algorithm, which represents the SLAM posterior in a sparse network known as thin junction trees (Pearl 1988; Cowell et al. 1999). The same idea was exploited by Frese (2004), who developed a similar tree factorization of the information matrix for efficient inference. Julier and Uhlmann (2000) developed a scalable technique called *covariance intersection*, which sparsely approximates the posterior in a way that provable prevents overconfidence. Their algorithm was successfully implemented on NASA's MARS Rover fleet (Uhlmann, Lanzagorta, and Julier 1999). The information filter perspective is also related to early work by Bulata and Devy (1996), whose approach acquired landmark models first in local landmark-centric reference frames, and only later assembles a consistent global map by resolving the relative information between landmarks. Another online filter related to this work is the SEIF algorithm, which was developed by Thrun et al. (2002). A greedy data association algorithm for SEIFs was developed by Liu and Thrun (2003), which was subsequently extended to multi-robot SLAM by Thrun and Liu (2003). A branch-and-bound data association search is due to Hähnel et al. (2003), based on earlier branch-and-bound methods by Lawler and Wood (1966) and Narendra and Fukunaga (1977). It parallels work by Kuipers et al. (2004), who developed a similar data association technique, albeit not in the context of an information theoretic concepts. Finally, certain 'offline' SLAM algorithms that solve the full SLAM problem, such as the ones by Bosse et al. (2004), Gutmann and Konolige (2000), and Frese (2004), have been shown to be fast enough to run online on limited-sized data sets. None of these approaches address how to incorporate occasional GPS measurements into SLAM.

3. Mapping SLAM Problems into Graphs

3.1. The Offline SLAM Problem

We begin our technical exposition with the basic notation used throughout this article. In SLAM, time is usually discrete, and t labels the time index. The robot pose at time t is denoted x_t; we will use $x_{1:t}$ to denote the set of poses from time 1 all the way to time t. The world itself is denoted m, where m is short for map. The map is assumed to be time-invariant, hence we do not use a time index to denote the map. In this paper, we think of the map of a (large) set of features m_j.

To acquire an environment map, the robot is able to sense. The measurement at time t is denoted z_t. Usually, the robot can sense multiple features at each point in time; hence each individual measurement beam is denoted z_t^i. Commonly, one assumes that z_t^i is a range measurement. The measurement function h describes how such a measurement comes into being:

$$z_t^i = h(x_t, m_j, i) + \varepsilon_t^i \tag{1}$$

here ε_t^i is a Gaussian random variable modeling the measurement noise, with zero mean and covariance Q_t, and m_j is the map feature sensed by the i-th measurement beam at time t. Put differently, we have

$$p(z_t^i \mid x_t, m) = \text{const.}\ \exp -\frac{1}{2}\ (z_t^i - h(x_t, m_j, i))^T$$
$$Q_t^{-1}\ (z_t^i - h(x_t, m_j, i)) \tag{2}$$

Some robotic systems are also are provided with a GPS system. Then the measurement is of the form

$$z_t^i = h(x_t, i) + \varepsilon_t^i \tag{3}$$

where z_t^i is a noisy estimate of the pose x_t, and ε_t^i is once again a Gaussian noise variable The mathematics for such measurements are analogous to those of nearby features; and GraphSLAM admits for arbitrary measurement functions h.

Finally, the robot changes its pose in SLAM by virtue of issuing control commands. The control asserted between time $t-1$ and time t is denoted u_t. The state transition is governed by the function g:

$$x_t = g(u_t, x_{t-1}) + \delta_t \tag{4}$$

where $\delta_t \sim \mathcal{N}(0, R_t)$ models the noise in the control command. The function g can be thought of as the kinematic model of the robot. Equation (4) induces the state transition probability

$$p(x_t \mid u_t, x_{t-1}) = \text{const.}\ \exp -\frac{1}{2}\ (x_t - g(u_t, x_{t-1}))^T$$
$$R_t^{-1}\ (x_t - g(u_t, x_{t-1})) \tag{5}$$

The offline SLAM posterior is now given by the following posterior probability over the robot path $x_{t:1}$ and the map m:

$$p(x_{1:t}, m \mid z_{1:t}, u_{1:t}) \tag{6}$$

This is the posterior probability over the entire path $x_{1:t}$ along with the map, instead of just the current pose x_t: We note that

in many SLAM problems, it suffices to determine the mode of this posterior. The actual posterior is usually too difficult to express for high-dimensional maps m, since it contains dependencies between any pair of features in m.

We note that a key assumption in our problem formulation is the assumption of independent Gaussian noise. GraphSLAM shares this assumption with the vast majority of published papers in the field of SLAM. The Gaussian noise assumption proves convenient in that it leads to a nice set of quadratic equations which can be solved efficiently. Other SLAM approaches have relaxed this assumption (Montemerlo et al. 2002) or made special provisions for incorporating non-Gaussian noise into Gaussian SLAM (Guivant and Masson 2005).

3.2. GraphSLAM: Basic Idea

Figure 1 illustrates the GraphSLAM algorithm. Shown there is the graph that GraphSLAM extracts from four poses labeled $x_1, \ldots, x_4$, and two map features m_1, m_2. Arcs in this graph come in two types: motion arcs and measurement arcs. Motion arcs link any two consecutive robot poses, and measurement arcs link poses to features that were measured there. Each edge in the graph corresponds to a nonlinear constraint. As we shall see later, these constraints represent the negative log likelihood of the measurement and the motion models, hence are best thought of as *information constraints*. Adding such a constraint to the graph is trivial for GraphSLAM; it involves no significant computation. The sum of all constraints results in a nonlinear *least squares problem*, as stated in Figure 1.

To compute a map posterior, GraphSLAM linearizes the set of constraints. The result of linearization is a sparse information matrix and an information vector. The sparseness of this matrix enables GraphSLAM to apply the variable elimination algorithm, thereby transforming the graph into a much smaller one only defined over robot poses. The path posterior map is then calculated using standard inference techniques. GraphSLAM also computes a map and certain marginal posteriors over the map; the full map posterior is of course quadratic in the size of the map and hence is usually not recovered.

3.3. Building Up the Graph

Suppose we are given a set of measurements $z_{1:t}$ with associated correspondence variables $c_{1:t}$, and a set of controls $u_{1:t}$. GraphSLAM turns this data into a graph. The nodes of this graph are the robot poses $x_{1:t}$ and the features in the map $m = \{m_j\}$. Each edge in the graph corresponds to an event: a motion event generates an edge between two robot poses, and a measurement event creates a link between a pose and a feature in the map. Edges represent soft constraints between poses and features in GraphSLAM.

For a linear system, these constraints are equivalent to entries in an information matrix and an information vector of a large system of equations. As usual, we will denote the information matrix by Ω and the information vector by ξ. As we shall see below, each measurement and each control leads to a *local* update of Ω and ξ, which corresponds to a local addition of an edge to the graph in GraphSLAM. In fact, the rule for incorporating a control or a measurement into Ω and ξ is a local addition, paying tribute to the important fact that information is an additive quantity.

Figure 2 illustrates the process of constructing the graph along with the corresponding information matrix. First consider a measurement z_t^i. This measurement provides information between the location of the feature $j = c_t^i$ and the robot pose x_t at time t. In GraphSLAM, this information is mapped into a constraint between x_t and m_j. We can think of this edge as a (possibly degenerate) "spring" in a spring-mass model. As we shall see below, the constraint is of the type:

$$(z_t^i - h(x_t, m_j, i))^T \; Q_t^{-1} \; (z_t^i - h(x_t, m_j, i)) \tag{7}$$

Here h is the measurement function, and Q_t is the covariance of the measurement noise. Figure 2(a) shows the addition of such a link into the graph maintained by GraphSLAM. Note that the constraint may be *degenerate*, that is, it may not constrain all dimensions of the robot pose x_t. This will be of no concern for the material yet to come.

In information form, the constraint is incorporated into Ω and ξ by adding values between the rows and columns connecting x_{t-1} and x_t. The magnitude of these values corresponds to the stiffness of the constraint, as governed by the uncertainty covariance Q_t of the motion model. This is illustrated in Figure 2(b), which shows the link between two robot poses along with the corresponding element in the information matrix.

Now consider robot motion. The control u_t provides information about the relative value of the robot pose at time $t-1$ and the pose at time t. Again, this information induces a constraint in the graph, which will be of the form:

$$(x_t - g(u_t, x_{t-1}))^T \; R_t^{-1} \; (x_t - g(u_t, x_{t-1})) \tag{8}$$

Here g is the kinematic motion model of the robot, and R_t is the covariance of the motion noise. Figure 2(b) illustrates the addition of such a link in the graph. It also shows the addition of a new element in the information matrix, between the pose x_t and the measurement z_t^i. This update is again additive. As before, the magnitude of these values reflects the residual uncertainty R_t due to the measurement noise; the less noisy the sensor, the larger the value added to Ω and ξ.

After incorporating all measurements $z_{1:t}$ and controls $u_{1:t}$, we obtain a sparse graph of soft constraints. The number of constraints in the graph is linear in the time elapsed, hence the graph is sparse. The sum of all constraints in the graph will be of the form

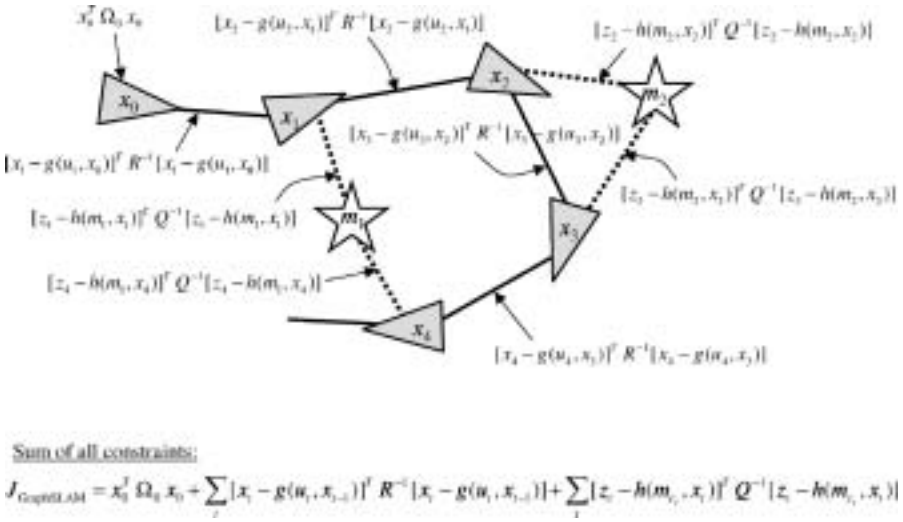


Fig. 1. GraphSLAM illustration, with 4 poses and two map features. Nodes in the graphs are robot poses and feature locations. The graph is populated by two types of edges: Solid edges which link consecutive robot poses, and dashed edges, which link poses with features sensed while the robot assumes that pose. Each link in GraphSLAM is a non-linear quadratic constraint. Motion constraints integrate the motion model; measurement constraints the measurement model. The target function of GraphSLAM is sum of these constraints. Minimizing it yields the most likely map and the most likely robot path.

$$
\begin{aligned}
J_{\text{GraphSLAM}} = x_0^T \; \Omega_0 \; x_0 \; + \; & \sum_t (x_t - g(u_t, x_{t-1}))^T \\
& R_t^{-1} \; (x_t - g(u_t, x_{t-1})) \\
+ \; & \sum_t \sum_i (z_t^i - h(y_t, c_t^i, i))^T \\
& Q_t^{-1} \; (z_t^i - h(y_t, c_t^i, i))
\end{aligned}
\tag{9}
$$

It is a function defined over pose variables $x_{1:t}$ and all feature locations in the map m. Notice that this expression also features an *anchoring constraint* of the form $x_0^T \; \Omega_0 \; x_0$. This constraint anchors the absolute coordinates of the map by initializing the very first pose of the robot as $(0 \;\; 0 \;\; 0)^T$.

In the associated information matrix Ω, the off-diagonal elements are all zero with two exceptions: between any two consecutive poses x_{t-1} and x_t will be a non-zero value that represents the information link introduced by the control u_t. Also non-zero will be any element between a map feature m_j and a pose x_t, if m_j was observed when the robot was at x_t. All elements between pairs of different features remain zero. This reflects the fact that we never receive information pertaining to their relative location—all we receive in SLAM are measurements that constrain the location of a feature relative to a robot pose. Thus, the information matrix is equally sparse; all but a linear number of its elements are zero.

3.4. Inference

Of course, neither the graph representation nor the information matrix representation gives us what we want: the map and the path. In GraphSLAM, the map and the path are obtained from the linearized information matrix Ω and the information vector ξ, via the equations $\Sigma = \Omega^{-1}$ and $\mu = \Sigma \; \xi$. This operation requires us to solve a system of linear equations. This raises the question on how efficiently we can recover the map estimate μ.

The answer to the complexity question depends on the topology of the world. If each feature is seen only locally in time, the graph represented by the constraints is linear. Thus, Ω can be reordered so that it becomes a band-diagonal matrix, that is, all non-zero values occur near its diagonal. The equation $\mu = \Omega^{-1}\xi$ can then be computed in linear time. This intuition carries over to a cycle-free world that is traversed once, so that each feature is seen for a short, consecutive period of time.

The more common case, however, involves features that are observed multiple times, with large time delays in between. This might be the case because the robot goes back and forth through a corridor, or because the world possesses *cycles*. In either situation, there will exist features m_j that are seen at drastically different time steps x_{t_1} and x_{t_2}, with $t_2 \gg t_1$. In our constraint graph, this introduces a cyclic dependence: x_{t_1} and x_{t_2} are linked through the sequence of controls $u_{t_1+1}, u_{t_1+2}, \ldots, u_{t_2}$ and through the joint observation links between x_{t_1} and m_j, and x_{t_2} and m_j, respectively. Such links make our variable reordering trick inapplicable, and recovering the map becomes more complex. In fact, since the inverse of Ω is multiplied with a vector, the result can be computed with optimization techniques such as conjugate gradient, without explicitly computing the full inverse matrix. Since most worlds possess cycles, this is the case of interest.

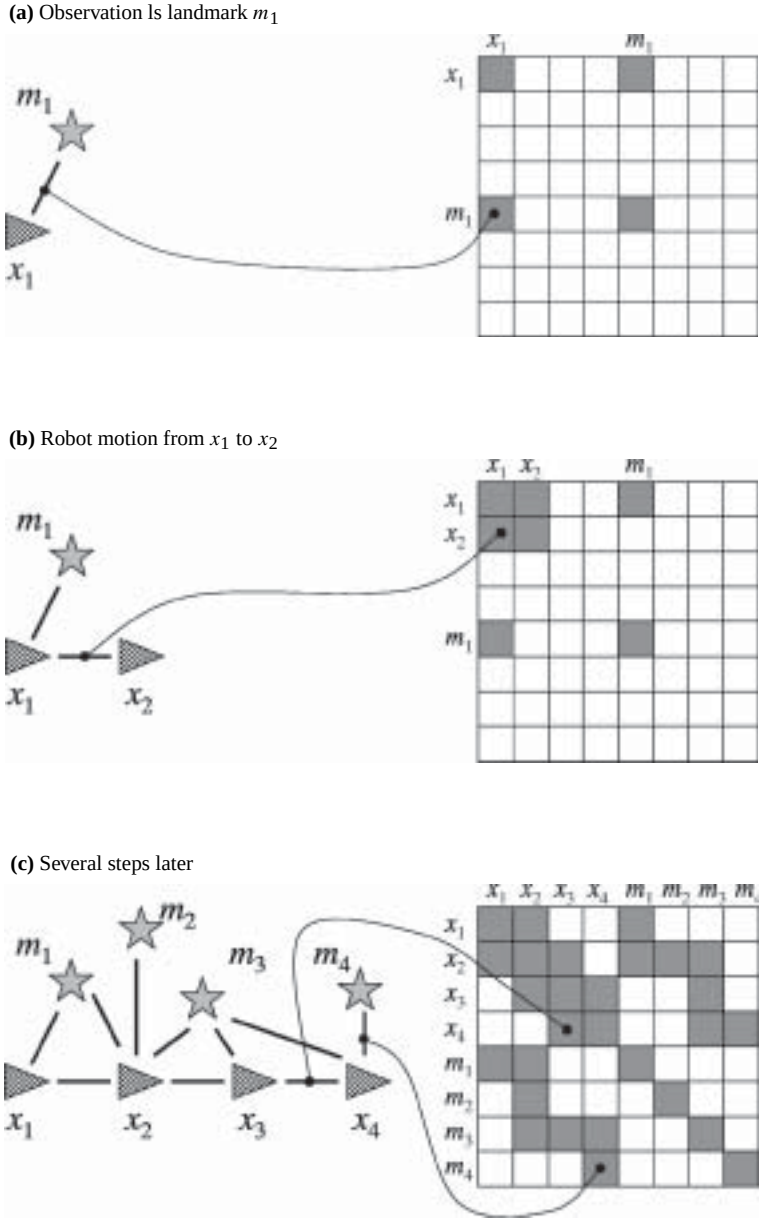
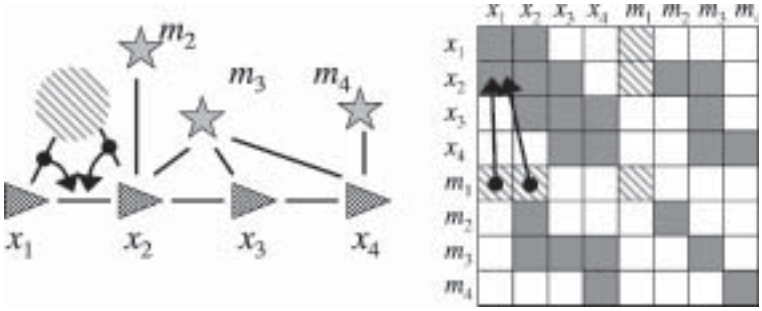


Fig. 2. Illustration of the acquisition of the information matrix in GraphSLAM. The left diagram shows the dependence graph, the right the information matrix.

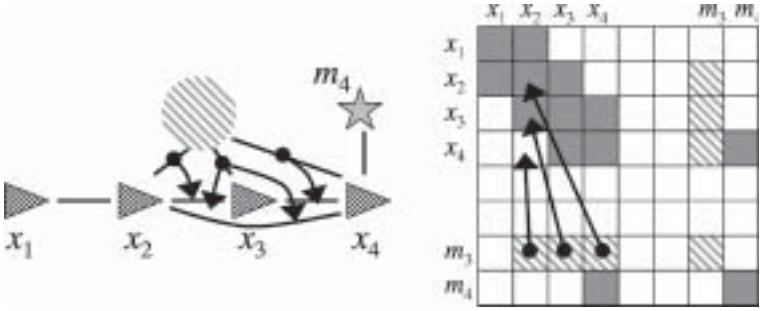
The GraphSLAM algorithm now employs an important *factorization trick*, which we can think of as propagating information trough the information matrix (in fact, it is a generalization of the well-known *variable elimination algorithm* for matrix inversion). Suppose we would like to remove a feature m_j from the information matrix Ω and the information state ξ. In our spring mass model, this is equivalent to removing the node and all springs attached to this node. As we shall see below, this is possible by a remarkably simple operation: we can remove all those springs between m_j and the poses at which m_j was observed, by introducing new springs between any pair of such poses.

This process is illustrated in Figure 3, which shows the removal of two map features, m_1 and m_3 (the removal of m_2 and m_4 is trivial in this example). In both cases, the feature removal modifies the link between any pair of poses from which a feature was originally observed. As illustrated in Figure 3(b), this operation may lead to the introduction of new

(a) The removal of m_1 changes the link between x_1 and x_2



(b) The removal of m_3 introduces a new link between x_2 and x_4



(c) Final result after removing all map features

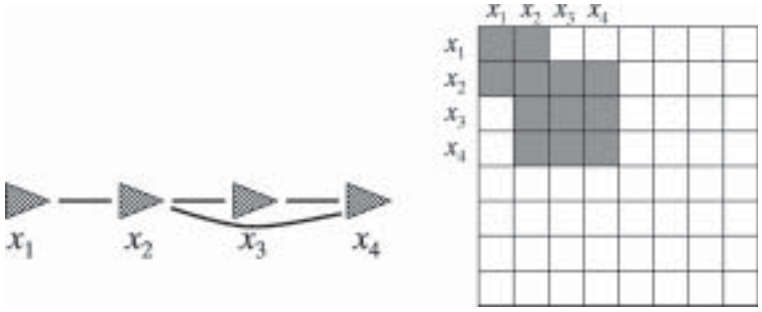


Fig. 3. Reducing the graph in GraphSLAM: Arcs are removed to yield a network of links that only connect robot poses.

links in the graph. In the example shown there, the removal of m_3 leads to a new link between x_2 and x_4.

Let $\tau(j)$ be the set of poses at which m_j was observed (that is: $x_t \in \tau(j) \Longleftrightarrow \exists i : c_t^i = j$). Then we already know that the feature m_j is only linked to poses x_t in $\tau(j)$; by construction, m_j is *not* linked to any other pose, or to any feature in the map. We can now set all links between m_j and the poses $\tau(j)$ to zero by introducing a new link between any two poses $x_t, x_{t'} \in \tau(j)$. Similarly, the information vector values for all poses $\tau(j)$ are also updated. An important characteristic of this operation is that it is local: It only involves a small number of constraints. After removing all links to m_j, we can safely remove m_j from the information matrix and vector. The resulting information matrix is smaller—it lacks an entry for m_j. However, it is equivalent for the remaining variables, in the sense that the posterior defined by this information matrix is mathematically equivalent to the original posterior before removing m_j. This equivalence is intuitive: We simply have replaced springs connecting m_j to various poses in our spring mass model by a set of springs directly linking these poses. In doing so, the total force asserted by these springs remains equivalent, with the only exception that m_j is now disconnected.

The virtue of this reduction step is that we can gradually transform our inference problem into a smaller one. By removing each feature m_j from Ω and ξ, we ultimately arrive at a much smaller information form $\tilde{\Omega}$ and $\tilde{\xi}$ defined only over the robot path variables. The reduction can be carried out in time linear in the size of the map; in fact, it generalizes the variable elimination technique for matrix inversion to the information form, in which we also maintain an information state. The posterior over the robot path is now recovered as $\tilde{\Sigma} = \tilde{\Omega}^{-1}$ and $\tilde{\mu} = \tilde{\Sigma}\xi$. Unfortunately, our reduction step does not eliminate cycles in the posterior. The remaining inference problem may still require more than linear time.

As a last step, GraphSLAM recovers the feature locations. Conceptually, this is achieved by building a new information matrix Ω_j and information vector ξ_j for each m_j. Both are defined over the variable m_j and the poses $\tau(j)$ at which m_j were observed. It contains the original links between m_j and $\tau(j)$, but the poses $\tau(j)$ are set to the values in $\tilde{\mu}$, without uncertainty. From this information form, it is now simple to calculate the location of m_j, using the common matrix inversion trick. Clearly, Ω_j contains only elements that connect to m_j; hence the inversion takes time linear in the number of poses in $\tau(j)$.

It should be apparent why the graph representation is such a natural representation. The full SLAM problem is solved by locally adding information into a large information graph, one edge at a time for each measurement z_t^i and each control u_t. To turn such information into an estimate of the map and the robot path, it is first linearized, then information between poses and features is gradually shifted to information between pairs of poses. The resulting structure only constrains the robot poses, which are then calculated using matrix inversion. Once the poses are recovered, the feature locations are calculated one after another, based on the original feature-to-pose information.

4. The GraphSLAM Algorithm

We will now make the various computational steps of the GraphSLAM precise. The full GraphSLAM algorithm will be described in a number of steps. The main difficulty in implementing the simple additive information algorithm pertains to the conversion of a conditional probability of the form $p(z_t^i \mid x_t, m)$ and $p(x_t \mid u_t, x_{t-1})$ into a link in the information matrix. The information matrix elements are all linear; hence this step involves linearizing $p(z_t^i \mid x_t, m)$ and $p(x_t \mid u_t, x_{t-1})$. To perform this linearization, we need an initial estimate $\mu_{0:t}$ for all poses $x_{0:t}$.

There exist a number of solutions to the problem of finding an initial mean μ suitable for linearization. For example, we can run an EKF SLAM and use its estimate for linearization (Dissanayake et al. 2001). GraphSLAM uses an even simpler technique: our initial estimate will simply be provided by chaining together the motion model $p(x_t \mid u_t, x_{t-1})$. Such an algorithm is outlined in Table 1, and called there **GraphSLAM_initialize**. This algorithm takes the controls $u_{1:t}$ as input, and outputs sequence of pose estimates $\mu_{0:t}$. It initializes the first pose by zero, and then calculates subsequent poses by recursively applying the velocity motion model. Since we are only interested in the mean poses vector $\mu_{0:t}$, **GraphSLAM_initialize** only uses the deterministic part of the motion model. It also does not consider any measurement in its estimation.

Once an initial $\mu_{0:t}$ is available, the GraphSLAM algorithm constructs the full SLAM information matrix Ω and the corresponding information vector ξ. This is achieved by linearizing the links in the graph. The algorithm **GraphSLAM_linearize** is depicted in Table 2. This algorithm contains a good amount of mathematical notation, much of which will become clear in our derivation of the algorithm further below. **GraphSLAM_linearize** accepts as an input the set of controls, $u_{1:t}$, the measurements $z_{1:t}$ and associated correspondence variables $c_{1:t}$, and the mean pose estimates $\mu_{0:t}$. It then gradually constructs the information matrix Ω and the information vector ξ through linearization, by locally adding sub-matrices in accordance with the information obtained from each measurement and each control.

In particular, line 2 in **GraphSLAM_linearize** initializes the information elements. The "infinite" information entry in line 3 fixes the initial pose x_0 to $(0\ 0\ 0)^T$. It is necessary, since otherwise the resulting matrix becomes singular, reflecting the fact that from relative information alone we cannot recover absolute estimates.

Controls are integrated in lines 4 through 9 of **GraphSLAM_linearize**. The pose $\hat{x}$ and the Jacobian G_t calculated

Table 1. Initialization of the Mean Pose Vector $\mu_{1:t}$ in the GraphSLAM Algorithm

Algorithm GraphSLAM_initialize($u_{1:t}$):

$$\begin{pmatrix} \mu_{0,x} \\ \mu_{0,y} \\ \mu_{0,\theta} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

for all controls $u_t = (v_t \;\; \omega_t)^T$ *do*

$$\begin{pmatrix} \mu_{t,x} \\ \mu_{t,y} \\ \mu_{t,\theta} \end{pmatrix} = \begin{pmatrix} \mu_{t-1,x} \\ \mu_{t-1,y} \\ \mu_{t-1,\theta} \end{pmatrix}$$

$$+ \begin{pmatrix} -\frac{v_t}{\omega_t} \sin \mu_{t-1,\theta} + \frac{v_t}{\omega_t} \sin(\mu_{t-1,\theta} + \omega_t \Delta t) \\ \frac{v_t}{\omega_t} \cos \mu_{t-1,\theta} - \frac{v_t}{\omega_t} \cos(\mu_{t-1,\theta} + \omega_t \Delta t) \\ \omega_t \Delta t \end{pmatrix}$$

endfor

return $\mu_{0:t}$

Table 2. Calculation of Ω and ξ in GraphSLAM

Algorithm GraphSLAM_linearize($u_{1:t}$, $z_{1:t}$, $c_{1:t}$, $\mu_{0:t}$):

set $\Omega = 0, \xi = 0$

add $\begin{pmatrix} \infty & 0 & 0 \\ 0 & \infty & 0 \\ 0 & 0 & \infty \end{pmatrix}$ *to* Ω *at* x_0

for all controls $u_t = (v_t \;\; \omega_t)^T$ *do*

$$\hat{x}_t = \mu_{t-1} + \begin{pmatrix} -\frac{v_t}{\omega_t} \sin \mu_{t-1,\theta} + \frac{v_t}{\omega_t} \sin(\mu_{t-1,\theta} + \omega_t \Delta t) \\ \frac{v_t}{\omega_t} \cos \mu_{t-1,\theta} - \frac{v_t}{\omega_t} \cos(\mu_{t-1,\theta} + \omega_t \Delta t) \\ \omega_t \Delta t \end{pmatrix}$$

$$G_t = \begin{pmatrix} 1 & 0 & \frac{v_t}{\omega_t} \cos \mu_{t-1,\theta} - \frac{v_t}{\omega_t} \cos(\mu_{t-1,\theta} + \omega_t \Delta t) \\ 0 & 1 & \frac{v_t}{\omega_t} \sin \mu_{t-1,\theta} - \frac{v_t}{\omega_t} \sin(\mu_{t-1,\theta} + \omega_t \Delta t) \\ 0 & 0 & 1 \end{pmatrix}$$

see next page for continuation

continued from the previous page

7: $\quad$ *add* $\begin{pmatrix} 1 \\ -G_t \end{pmatrix} R_t^{-1} \begin{pmatrix} 1 & -G_t \end{pmatrix}$ *to* Ω *at* x_t *and* x_{t-1}

8: $\quad$ *add* $\begin{pmatrix} 1 \\ -G_t \end{pmatrix} R_t^{-1} [\hat{x}_t + G_t\, \mu_{t-1}]$ *to* ξ *at* x_t *and* x_{t-1}

9: *endfor*

10: *for all measurements* z_t *do*

11: $Q_t = \begin{pmatrix} \sigma_r & 0 & 0 \\ 0 & \sigma_\phi & 0 \\ 0 & 0 & \sigma_s \end{pmatrix}$

12: $\quad$ *for all observed features* $z_t^i = (r_t^i \;\; \phi_t^i)^T$ *do*

13: $\quad\quad j = c_t^i$

14: $\quad\quad \delta = \begin{pmatrix} \delta_x \\ \delta_y \end{pmatrix} = \begin{pmatrix} \mu_{j,x} - \mu_{t,x} \\ \mu_{j,y} - \mu_{t,y} \end{pmatrix}$

15: $\quad\quad q = \delta^T \delta$

16: $\quad\quad \hat{z}_t^i = \begin{pmatrix} \sqrt{q} \\ \text{atan2}(\delta_y, \delta_x) - \mu_{t,\theta} \end{pmatrix}$

17: $\quad\quad H_t^i = \frac{1}{q} \begin{pmatrix} \sqrt{q}\delta_x & -\sqrt{q}\delta_y & 0 & -\sqrt{q}\delta_x & \sqrt{q}\delta_y \\ \delta_y & \delta_x & -1 & -\delta_y & -\delta_x \end{pmatrix}$

18: $\quad\quad$ *add* $H_t^{iT}\; Q_t^{-1}\; H_t^i$ *to* Ω *at* x_t *and* m_j

19: $\quad\quad$ *add* $H_t^{iT}\; Q_t^{-1}\; [\; z_t^i - \hat{z}_t^i - H_t^i \begin{pmatrix} \mu_{t,x} \\ \mu_{t,y} \\ \mu_{t,\theta} \\ \mu_{j,x} \\ \mu_{j,y} \end{pmatrix}]$ *to* ξ *at* x_t *and* m_j

20: $\quad$ *endfor*

21: *endfor*

22: *return* Ω, ξ

in lines 5 and 6 represent the linear approximation of the non-linear measurement function g. As is obvious from these equations, this linearization step utilizes the pose estimates $\mu_{0:t-1}$, with $\mu_0 = (0 \;\; 0 \;\; 0)^T$. This leads to the updates for Ω, and ξ, calculated in lines 7, and 8, respectively. Both terms are added into the corresponding rows and columns of Ω and ξ. This addition realizes the inclusion of a new constraint into the SLAM posterior, very much along the lines of the intuitive description in the previous section.

Measurements are integrated in lines 10 through 21 of **GraphSLAM_linearize**. The matrix Q_t calculated in line 11 is the familiar measurement noise covariance. Lines 13 through 17 compute the Taylor expansion of the measurement function. This calculation assumes *known* correspondence between observed features and features in the map (line 13). Attention has to be paid to the implementation of line 16, since the angular expressions can be shifted arbitrarily by 2π. This calculation culminates in the computation of the measurement update in lines 18 and 19. The matrix that is being added to Ω in line 18 is of dimension 5×5. To add it, we decompose it into a matrix of dimension 3×3 for the pose x_t, a matrix of dimension 2×2 for the feature m_j, and two matrices of dimension 3×2 and 2×3 for the link between x_t and m_j. Those are added to Ω at the corresponding rows and columns. Similarly, the vector added to the information vector ξ is of vertical dimension 5. It is also chopped into two vectors of size 3 and 2, and added to the elements corresponding to x_t and m_j, respectively. The result of **GraphSLAM_linearize**

is an information vector ξ and a matrix Ω. We already noted that Ω is sparse. It contains only non-zero sub-matrices along the main diagonal, between subsequent poses, and between poses and features in the map. The running time of this algorithm is linear in t, the number of time steps at which data was accrued.

The next step of the GraphSLAM algorithm pertains to reducing the dimensionality of the information matrix/vector. This is achieved through the algorithm **GraphSLAM_reduce** in Table 3. This algorithm takes as input Ω and ξ defined over the full space of map features and poses, and outputs a reduced matrix $\tilde{\Omega}$ and vectors $\tilde{\xi}$ defined over the space of all poses (but not the map!). This transformation is achieved by removing features m_j one at a time, in lines 4 through 9 of **GraphSLAM_reduce**. The bookkeeping of the exact indexes of each item in $\tilde{\Omega}$ and $\tilde{\xi}$ is a bit tedious, hence Table 3 only provides an intuitive account.

Line 5 calculates the set of poses $\tau(j)$ at which the robot observed feature j. It then extracts two sub-matrices from the present $\tilde{\Omega}$: $\tilde{\Omega}_{j,j}$ and $\tilde{\Omega}_{\tau(j),j}$. $\tilde{\Omega}_{j,j}$ is the quadratic sub-matrix between m_j and m_j, and $\tilde{\Omega}_{\tau(j),j}$ is composed of the off-diagonal elements between m_j and the pose variables $\tau(j)$. It also extracts from the information state vector $\tilde{\xi}$ the elements corresponding to the j-th feature, denoted here as ξ_j. It then subtracts information from $\tilde{\Omega}$ and $\tilde{\xi}$ as stated in lines 6 and 7. After this operation, the rows and columns for the feature m_j are zero. These rows and columns are then removed, reducing the dimension on $\tilde{\Omega}$ and $\tilde{\xi}$ accordingly. This process is iterated until all features have been removed, and only pose variables remain in $\tilde{\Omega}$ and $\tilde{\xi}$. The complexity of **GraphSLAM_reduce** is once again linear in t.

The last step in the GraphSLAM algorithm computes the mean and covariance for all poses in the robot path, and a mean location estimate for all features in the map. This is achieved through **GraphSLAM_solve** in Table 4. Line 3 computes the path estimates $\mu_{0:t}$. This can be achieved by inverting the reduced information matrix $\tilde{\Omega}$ and multiplying the resulting covariance with the information vector, or by optimization techniques such as conjugate gradient descent. Subsequently, **GraphSLAM_solve** computes the location of each feature in lines 4 through 7. The return value of **GraphSLAM_solve** contains the mean for the robot path and all features in the map, but only the covariance for the robot path.

The quality of the solution calculated by the GraphSLAM algorithm depends on the goodness of the initial mean estimates, calculated by **GraphSLAM_initialize**. The x- and y-components of these estimates affect the respective models in a linear way, hence the linearization does not depend on these values. Not so for the orientation variables in $\mu_{0:t}$. Errors in these initial estimates affect the accuracy of the Taylor approximation, which in turn affects the result.

To reduce potential errors due to the Taylor approximation in the linearization, the procedures **GraphSLAM_linearize**, **GraphSLAM_reduce**, and **GraphSLAM_solve** are run multiple times over the same data set. Each iteration takes as an input an estimated mean vector $\mu_{0:t}$ from the previous iteration, and outputs a new, improved estimate. The iterations of the GraphSLAM optimization are only necessary when the initial pose estimates have high error (e.g. more than 20 degrees orientation error). A small number of iterations (e.g. 3) is usually sufficient.

Table 5 summarizes the resulting algorithm. It initializes the means, then repeats the construction step, the reduction step, and the solution step. Typically, two or three iterations suffice for convergence. The resulting mean μ is our best guess of the robot's path and the map.

5. Mathematical Derivation of GraphSLAM

The derivation of the GraphSLAM algorithm begins with a derivation of a recursive formula for calculating the full SLAM posterior, represented in information form. We then investigate each term in this posterior, and derive from them the additive SLAM updates through Taylor expansions. From that, we will derive the necessary equations for recovering the path and the map.

5.1. The Full SLAM Posterior

It will be beneficial to introduce a variable for the augmented state of the full SLAM problem. We will use y to denote state variables that combine one or more poses x with the map m. In particular, we define $y_{0:t}$ to be a vector composed of the path $x_{0:t}$ and the map m, whereas y_t is composed of the momentary pose at time t and the map m:

$$y_{0:t} = \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_t \\ m \end{pmatrix} \quad \text{and} \quad y_t = \begin{pmatrix} x_t \\ m \end{pmatrix} \tag{10}$$

The posterior in the full SLAM problem is $p(y_{0:t} \mid z_{1:t}, u_{1:t}, c_{1:t})$, where $z_{1:t}$ are the familiar measurements with correspondences $c_{1:t}$, and $u_{1:t}$ are the controls. Bayes rule enables us to factor this posterior:

$$\begin{aligned} & p(y_{0:t} \mid z_{1:t}, u_{1:t}, c_{1:t}) \\ &= \eta\ p(z_t \mid y_{0:t}, z_{1:t-1}, u_{1:t}, c_{1:t})\ p(y_{0:t} \mid z_{1:t-1}, u_{1:t}, c_{1:t}) \end{aligned} \tag{11}$$

where η is the familiar normalizer. The first probability on the right-hand side can be reduced by dropping irrelevant conditioning variables:

$$p(z_t \mid y_{0:t}, z_{1:t-1}, u_{1:t}, c_{1:t}) = p(z_t \mid y_t, c_t) \tag{12}$$

Table 3. Algorithm for Reducing the Size of the Information Representation of the Posterior in GraphSLAM

Algorithm GraphSLAM_reduce(Ω, ξ):

$\tilde{\Omega} = \Omega$

$\tilde{\xi} = \xi$

for each feature j *do*

  let $\tau(j)$ *be the set of all poses* x_t *at which* j *was observed*

  subtract $\tilde{\Omega}_{\tau(j),j}\ \tilde{\Omega}_{j,j}^{-1}\ \xi_j$ *from* $\tilde{\xi}$ *at* $x_{\tau(j)}$ *and* m_j

  subtract $\tilde{\Omega}_{\tau(j),j}\ \tilde{\Omega}_{j,j}^{-1}\ \tilde{\Omega}_{j,\tau(j)}$ *from* $\tilde{\Omega}$ *at* $x_{\tau(j)}$ *and* m_j

  remove from $\tilde{\Omega}$ *and* $\tilde{\xi}$ *all rows/columns corresponding to* j

endfor

return $\tilde{\Omega}, \tilde{\xi}$

Table 4. Algorithm for Updating the Posterior μ

Algorithm GraphSLAM_solve($\tilde{\Omega}, \tilde{\xi}, \Omega, \xi$):

$\Sigma_{0:t} = \tilde{\Omega}^{-1}$

$\mu_{0:t} = \Sigma_{0:t}\ \tilde{\xi}$

for each feature j *do*

  set $\tau(j)$ *to the set of all poses* x_t *at which* j *was observed*

  $\mu_j = \Omega_{j,j}^{-1}\ (\xi_j + \Omega_{j,\tau(j)}\ \tilde{\mu}_{\tau(j)})$

endfor

return $\mu, \Sigma_{0:t}$

Table 5. The GraphSLAM Algorithm for the Full SLAM Problem with Known Correspondence

Algorithm GraphSLAM_known_correspondence($u_{1:t}, z_{1:t}, c_{1:t}$):

$\mu_{0:t} =$ **GraphSLAM_initialize**$(u_{1:t})$

repeat

  $\Omega, \xi =$ **GraphSLAM_linearize**$(u_{1:t}, z_{1:t}, c_{1:t}, \mu_{0:t})$

  $\tilde{\Omega}, \tilde{\xi} =$ **GraphSLAM_reduce**(Ω, ξ)

  $\mu, \Sigma_{0:t} =$ **GraphSLAM_solve**$(\tilde{\Omega}, \tilde{\xi}, \Omega, \xi)$

until convergence

return μ

Similarly, we can factor the second probability by partitioning $y_{0:t}$ into x_t and $y_{0:t-1}$, and obtain:

$$\begin{aligned} & p(y_{0:t} \mid z_{1:t-1}, u_{1:t}, c_{1:t}) \qquad (13) \\ = \;& p(x_t \mid y_{0:t-1}, z_{1:t-1}, u_{1:t}, c_{1:t})\; p(y_{0:t-1} \mid z_{1:t-1}, u_{1:t}, c_{1:t}) \\ = \;& p(x_t \mid x_{t-1}, u_t)\; p(y_{0:t-1} \mid z_{1:t-1}, u_{1:t-1}, c_{1:t-1}) \end{aligned}$$

Putting these expressions back into (11) gives us the recursive definition of the full SLAM posterior:

$$\begin{aligned} & p(y_{0:t} \mid z_{1:t}, u_{1:t}, c_{1:t}) \qquad (14) \\ = \;& \eta\; p(z_t \mid y_t, c_t) \\ & p(x_t \mid x_{t-1}, u_t)\; p(y_{0:t-1} \mid z_{1:t-1}, u_{1:t-1}, c_{1:t-1}) \end{aligned}$$

The closed form expression is obtained through induction over t. Here $p(y_0)$ is the prior over the map m and the initial pose x_0.

$$\begin{aligned} & p(y_{0:t} \mid z_{1:t}, u_{1:t}, c_{1:t}) \qquad (15) \\ & = \eta\; p(y_0) \prod_t p(x_t \mid x_{t-1}, u_t)\; p(z_t \mid y_t, c_t) \\ & = \eta\; p(y_0) \prod_t \left[p(x_t \mid x_{t-1}, u_t) \prod_i p(z_t^i \mid y_t, c_t^i) \right] \end{aligned}$$

Here, as before, z_t^i is the i-th measurement in the measurement vector z_t at time t. The prior $p(y_0)$ factors into two independent priors, $p(x_0)$ and $p(m)$. In SLAM, we usually have no prior knowledge about the map m. We simply replace $p(y_0)$ by $p(x_0)$ and subsume the factor $p(m)$ into the normalizer η.

5.2. The Negative Log Posterior

The information form represents probabilities in logarithmic form. The log-SLAM posterior follows directly from the previous equation:

$$\begin{aligned} & \log p(y_{0:t} \mid z_{1:t}, u_{1:t}, c_{1:t}) \qquad (16) \\ & = \text{const.} + \log p(x_0) \\ & + \sum_t \left[\log p(x_t \mid x_{t-1}, u_t) + \sum_i \log p(z_t^i \mid y_t, c_t^i) \right] \end{aligned}$$

As stated above, we assume the outcome of robot motion is distributed normally according to $\mathcal{N}(g(u_t, x_{t-1}), R_t)$, where g is the deterministic motion function, and R_t is the covariance of the motion error. Likewise, measurements z_t^i are generated according to $\mathcal{N}(h(y_t, c_t^i, i), Q_t)$, where h is the familiar measurement function and Q_t is the measurement error covariance. In equations, we have:

$$p(x_t \mid x_{t-1}, u_t) \qquad (17)$$
$$= \eta\; \exp\left\{ -\frac{1}{2} (x_t - g(u_t, x_{t-1}))^T\; R_t^{-1}\; (x_t - g(u_t, x_{t-1})) \right\}$$

$$p(z_t^i \mid y_t, c_t^i) \qquad (18)$$
$$= \eta\; \exp\left\{ -\frac{1}{2} (z_t^i - h(y_t, c_t^i, i))^T\; Q_t^{-1}\; (z_t^i - h(y_t, c_t^i, i)) \right\}$$

The prior $p(x_0)$ in (16) is also easily expressed by a Gaussian-type distribution. It *anchors* the initial pose x_0 to the origin of the global coordinate system: $x_0 = (0\ 0\ 0)^T$:

$$p(x_0) \;=\; \eta\; \exp\left\{ -\frac{1}{2}\; x_0^T\; \Omega_0\; x_0 \right\} \qquad (19)$$

with

$$\Omega_0 \;=\; \begin{pmatrix} \infty & 0 & 0 \\ 0 & \infty & 0 \\ 0 & 0 & \infty \end{pmatrix} \qquad (20)$$

For now, it does not concern us that the value of ∞ cannot be implemented, as we can easily substitute ∞ with a large positive number. This leads to the following quadratic form of the negative log-SLAM posterior in (16):

$$\begin{aligned} & -\log p(y_{0:t} \mid z_{1:t}, u_{1:t}, c_{1:t}) \qquad (21) \\ = \;& \text{const.} + \frac{1}{2} \Bigg[x_0^T\; \Omega_0\; x_0 + \sum_t (x_t - g(u_t, x_{t-1}))^T \\ & R_t^{-1}\; (x_t - g(u_t, x_{t-1})) + \sum_t \sum_i (z_t^i - h(y_t, c_t^i, i))^T \\ & Q_t^{-1}\; (z_t^i - h(y_t, c_t^i, i)) \Bigg] \end{aligned}$$

This is essentially the same as $J_{\text{GraphSLAM}}$ in eq. (9), with a few differences pertaining to the omission of normalization constants (including a multiplication with -1). Equation (21) highlights an essential characteristic of the full SLAM posterior in the information form: it is composed of a number of quadratic terms, one for the prior, and one for each control and each measurement.

5.3. Taylor Expansion

The various terms in eq. (21) are quadratic in the functions g and h, not in the variables we seek to estimate (poses and the map). GraphSLAM alleviates this problem by *linearizing* g and h via Taylor expansion. In particular, we have:

$$g(u_t, x_{t-1}) \approx g(u_t, \mu_{t-1}) + \underbrace{g'(u_t, \mu_{t-1})}_{=:\, G_t} (x_{t-1} - \mu_{t-1}) \qquad (22)$$

$$h(y_t, c_t^i, i) \approx h(\mu_t, c_t^i, i) \;+\; \underbrace{h'(\mu_t)}_{=:\, H_t^i} (y_t - \mu_t) \qquad (23)$$

Here μ_t is the current estimate of the state vector y_t, and $H_t^i = h_t^i \; F_{x,j}$ for the projection matrix $F_{x,j}$ as indicated.

This linear approximation turns the log-likelihood (21) into a function that is quadratic in $y_{0:t}$. In particular, we obtain:

$$\begin{aligned}\log p(y_{0:t} \mid z_{1:t}, u_{1:t}, c_{1:t}) \;\; &= \;\; \text{const.} - \frac{1}{2} \\ &\Big\{ x_0^T \; \Omega_0 \; x_0 + \sum_t [x_t - g(u_t, \mu_{t-1}) - G_t(x_{t-1} - \mu_{t-1})]^T \\ &\qquad R_t^{-1} \; [x_t - g(u_t, \mu_{t-1}) - G_t(x_{t-1} - \mu_{t-1})] \\ &+ \sum_i [z_t^i - h(\mu_t, c_t^i, i) - H_t^i (y_t - \mu_t)]^T \\ &\qquad Q_t^{-1} \; [z_t^i - h(\mu_t, c_t^i, i) - H_t^i (y_t - \mu_t)] \Big\}\end{aligned} \tag{24}$$

This function is indeed a quadratic in $y_{0:t}$, and it is convenient to reorder its terms, omitting several constant terms.

$$\begin{aligned}\log p(y_{0:t} \mid z_{1:t}, u_{1:t}, c_{1:t}) \;\; &= \;\; \text{const.} \\ &- \frac{1}{2} \; \underbrace{x_0^T \; \Omega_0 \; x_0}_{\text{quadratic in } x_0} \\ &- \frac{1}{2} \sum_t \underbrace{x_{t-1:t}^T \begin{pmatrix} 1 \\ -G_t \end{pmatrix} R_t^{-1} \; (1 \;\; -G_t) \; x_{t-1:t}}_{\text{quadratic in } x_{t-1:t}} \\ &\qquad + \underbrace{x_{t-1:t}^T \begin{pmatrix} 1 \\ -G_t \end{pmatrix} R_t^{-1} \; [g(u_t, \mu_{t-1}) + G_t \; \mu_{t-1}]}_{\text{linear in } x_{t-1:t}} \\ &- \frac{1}{2} \sum_i \underbrace{y_t^T \; H_t^{iT} \; Q_t^{-1} \; H_t^i \; y_t}_{\text{quadratic in } y_t} \\ &+ \underbrace{y_t^T \; H_t^{iT} \; Q_t^{-1} \; [z_t^i - h(\mu_t, c_t^i, i) - H_t^i \mu_t]}_{\text{linear in } y_t}\end{aligned} \tag{25}$$

Here $x_{t-1:t}$ denotes the vector concatenating x_{t-1} and x_t; hence we can write $(x_t - G_t \; x_{t-1})^T = x_{t-1:t}^T \; (1 \;\; -G_t)^T$. If we collect all quadratic terms into the matrix Ω, and all linear terms into a vector ξ, we see that expression (24) is of the form:

$$\log p(y_{0:t} \mid z_{1:t}, u_{1:t}, c_{1:t}) = \text{const.} - \frac{1}{2} \; y_{0:t}^T \; \Omega \; y_{0:t} + y_{0:t}^T \; \xi \tag{26}$$

5.4. Constructing the Information Form

We can read off these terms directly from (25), and verify that they are indeed implemented in the algorithm **GraphSLAM_linearize** in Table 2:

- **Prior.** The initial pose prior manifests itself by a quadratic term Ω_0 over the initial pose variable x_0 in the information matrix. Assuming appropriate extension of the matrix Ω_0 to match the dimension of $y_{0:t}$, we have:

$$\Omega \;\; \longleftarrow \;\; \Omega_0 \tag{27}$$

  This initialization is performed in lines 2 and 3 of the algorithm **GraphSLAM_linearize**.

- **Controls.** From (25), we see that each control u_t adds to Ω and ξ the following terms, assuming that the matrices are rearranged so as to be of matching dimensions:

$$\Omega \longleftarrow \Omega + \begin{pmatrix} 1 \\ -G_t \end{pmatrix} R_t^{-1} \; (1 \;\; -G_t) \tag{28}$$

$$\xi \longleftarrow \xi + \begin{pmatrix} 1 \\ -G_t \end{pmatrix} R_t^{-1} \; [g(u_t, \mu_{t-1}) + G_t \; \mu_{t-1}] \tag{29}$$

  This is realized in lines 4 through 9 in **GraphSLAM_linearize**.

- **Measurements.** According to eq. (25), each measurement z_t^i transforms Ω and ξ by adding the following terms, once again assuming appropriate adjustment of the matrix dimensions:

$$\Omega \longleftarrow \Omega + H_t^{iT} \; Q_t^{-1} \; H_t^i \tag{30}$$

$$\xi \longleftarrow \xi + H_t^{iT} \; Q_t^{-1} \; [z_t^i - h(\mu_t, c_t^i, i) - H_t^i \mu_t] \tag{31}$$

  This update occurs in lines 10 through 21 in **GraphSLAM_linearize**.

This proves the correctness of the construction algorithm **GraphSLAM_linearize**, relative to our Taylor expansion approximation.

We also note that the steps above only affect off-diagonal elements that involve at least one pose. Thus, all between-feature elements are zero in the resulting information matrix.

5.5. Reducing the Information Form

The reduction step **GraphSLAM_reduce** is based on a factorization of the full SLAM posterior.

$$\begin{aligned}p(y_{0:t} \mid z_{1:t}, u_{1:t}, c_{1:t}) = \; &p(x_{0:t} \mid z_{1:t}, u_{1:t}, c_{1:t}) \\ &p(m \mid x_{0:t}, z_{1:t}, u_{1:t}, c_{1:t})\end{aligned} \tag{32}$$

Here $p(x_{0:t} \mid z_{1:t}, u_{1:t}, c_{1:t}) \sim \mathcal{N}(\Omega, \xi)$ is the posterior over paths alone, with the map integrated out:

$$p(x_{0:t} \mid z_{1:t}, u_{1:t}, c_{1:t}) = \int p(y_{0:t} \mid z_{1:t}, u_{1:t}, c_{1:t}) \; dm \tag{33}$$

As we will show shortly, this probability is indeed calculated by the algorithm **GraphSLAM_reduce** in Table 3, since

$$p(x_{0:t} \mid z_{1:t}, u_{1:t}, c_{1:t}) \;\; \sim \;\; \mathcal{N}(\tilde{\xi}, \tilde{\Omega}) \tag{34}$$

In general, the integration in (33) will be intractable, due to the large number of variables in m. For Gaussians, this integral can be calculated in closed form. The key insight is given by the *marginalization lemma* for Gaussians, stated in Table 7.

Table 6. The (Specialized) Inversion Lemma, Sometimes Called the *Sherman/Morrison Formula* (see the Appendix for a derivation)

Inversion Lemma. For any invertible quadratic matrices R and Q and any matrix P with appropriate dimensions, the following holds true

$$(R + P\ Q\ P^T)^{-1} \quad = \quad R^{-1} - R^{-1}\ P\ (Q^{-1} + P^T\ R^{-1}\ P)^{-1}\ P^T\ R^{-1}$$

assuming that all above matrices can be inverted as stated.

Table 7. Lemma for Marginalizing Gaussians in Information Form. The Form of the Covariance $\bar{\Omega}_{xx}$ in This Lemma is Also as *Schur complement* (a derivation can be found in the Appendix).

Marginals of a multivariate Gaussian. Let the probability distribution $p(x, y)$ over the random vectors x and y be a Gaussian represented in the information form

$$\Omega \quad = \quad \begin{pmatrix} \Omega_{xx} & \Omega_{xy} \\ \Omega_{yx} & \Omega_{yy} \end{pmatrix} \qquad \text{and} \qquad \xi \quad = \quad \begin{pmatrix} \xi_x \\ \xi_y \end{pmatrix}$$

If Ω_{yy} is invertible, the marginal $p(x)$ is a Gaussian whose information representation is

$$\bar{\Omega}_{xx} \quad = \quad \Omega_{xx} - \Omega_{xy}\ \Omega_{yy}^{-1}\ \Omega_{yx} \qquad \text{and} \qquad \bar{\xi}_x \quad = \quad \xi_x - \Omega_{xy}\ \Omega_{yy}^{-1}\ \xi_y$$

Let us subdivide the matrix Ω and the vector ξ into sub-matrices, for the robot path $x_{0:t}$ and the map m:

$$\Omega \quad = \quad \begin{pmatrix} \Omega_{x_{0:t},x_{0:t}} & \Omega_{x_{0:t},m} \\ \Omega_{m,x_{0:t}} & \Omega_{m,m} \end{pmatrix} \tag{35}$$

$$\xi \quad = \quad \begin{pmatrix} \xi_{x_{0:t}} \\ \xi_m \end{pmatrix} \tag{36}$$

According to the *marginalization lemma*, the probability (34) is obtained as

$$\tilde{\Omega} \quad = \quad \Omega_{x_{0:t},x_{0:t}} - \Omega_{x_{0:t},m}\ \Omega_{m,m}^{-1}\ \emptyset mega_{m,x_{0:t}} \tag{37}$$

$$\tilde{\xi} \quad = \quad \xi_{x_{0:t}} - \Omega_{x_{0:t},m}\ \Omega_{m,m}^{-1}\ \xi_m \tag{38}$$

The matrix $\Omega_{m,m}$ is block-diagonal. This follows from the way Ω is constructed, in particular the absence of any links between pairs of features. This makes the inversion efficient:

$$\Omega_{m,m}^{-1} \quad = \quad \sum_j F_j^T\ \Omega_{j,j}^{-1}\ F_j \tag{39}$$

where $\Omega_{j,j} = F_j \Omega F_j^T$ is the sub-matrix of Ω that corresponds to the j-th feature in the map, that is

$$F_j \quad = \quad \begin{pmatrix} 0 \cdots 0 & \underbrace{1\ 0}_{j-\text{th feature}} & 0 \cdots 0 \\ 0 \cdots 0 & \underbrace{0\ 1}_{j-\text{th feature}} & 0 \cdots 0 \end{pmatrix} \tag{40}$$

This insight makes it possible to decompose the implement eqs (37) and (38) into a sequential update:

$$\tilde{\Omega} \quad = \quad \Omega_{x_{0:t},x_{0:t}} - \sum_j \Omega_{x_{0:t},j}\ \Omega_{j,j}^{-1}\ ;\ \Omega_{j,x_{0:t}} \tag{41}$$

$$\tilde{\xi} \quad = \quad \xi_{x_{0:t}} - \sum_j \Omega_{x_{0:t},j}\ \Omega_{j,j}^{-1}\ \xi_j \tag{42}$$

The matrix $\Omega_{x_{0:t},j}$ is non-zero only for elements in $\tau(j)$, the set of poses at which feature j was observed. This essentially proves the correctness of the reduction algorithm **GraphSLAM_reduce** in Table 3. The operation performed on Ω in this algorithm can be thought of as the variable elimination algorithm for matrix inversion, applied to the feature variables but not the robot pose variables.

5.6. Recovering the Path and the Map

The algorithm **GraphSLAM_solve** in Table 4 calculates the mean and variance of the Gaussian $\mathcal{N}(\tilde{\xi}, \tilde{\Omega})$:

$$\tilde{\Sigma} \quad = \quad \tilde{\Omega}^{-1} \tag{43}$$

$$\tilde{\mu} \quad = \quad \tilde{\Sigma}\ \tilde{\xi} \tag{44}$$

In particular, this operation provides us with the mean of the posterior on the robot path; it does not give us the locations of the features in the map.

It remains to recover the second factor of eq. (32):

$$p(m \mid x_{0:t}, z_{1:t}, u_{1:t}, c_{1:t}) \tag{45}$$

Table 8. Lemma for Conditioning Gaussians in Information Form (a derivation can be found in the Appendix)

Conditionals of a multivariate Gaussian. Let the probability distribution $p(x, y)$ over the random vectors x and y be a Gaussian represented in the information form

$$\Omega = \begin{pmatrix} \Omega_{xx} & \Omega_{xy} \\ \Omega_{yx} & \Omega_{yy} \end{pmatrix} \quad \text{and} \quad \xi = \begin{pmatrix} \xi_x \\ \xi_y \end{pmatrix}$$

The conditional $p(x \mid y)$ is a Gaussian with information matrix Ω_{xx} and information vector $\xi_x + \Omega_{xy}\, y$.

The *conditioning lemma*, stated in Table 8, shows that this probability distribution is Gaussian with the parameters

$$\Sigma_m = \Omega^{-1}_{m,m} \tag{46}$$

$$\mu_m = \Sigma_m(\xi_m + \Omega_{m,x_{0:t}}\tilde{\xi}) \tag{47}$$

Here ξ_m and $\Omega_{m,m}$ are the sub-vector of ξ, and the sub-matrix of Ω, respectively, restricted to the map variables. The matrix $\Omega_{m,x_{0:t}}$ is the off-diagonal sub-matrix of Ω that connects the robot path to the map. As noted before, $\Omega_{m,m}$ is block-diagonal, hence we can decompose

$$p(m \mid x_{0:t}, z_{1:t}, u_{1:t}, c_{1:t}) = \prod_j p(m_j \mid x_{0:t}, z_{1:t}, u_{1:t}, c_{1:t}) \tag{48}$$

where each $p(m_j \mid x_{0:t}, z_{1:t}, u_{1:t}, c_{1:t})$ is distributed according to

$$\Sigma_j = \Omega^{-1}_{j,j} \tag{49}$$

$$\mu_j = \Sigma_j(\xi_j + \Omega_{j,x_{0:t}}\tilde{\mu}) = \Sigma_j(\xi_j + \Omega_{j,\tau(j)}\tilde{\mu}_{\tau(j)}) \tag{50}$$

The last transformation exploited the fact that the sub-matrix $\Omega_{j,x_{0:t}}$ is zero except for those pose variables $\tau(j)$ from which the j-th feature was observed.

It is important to notice that this is a Gaussian $p(m \mid x_{0:t}, z_{1:t}, u_{1:t}, c_{1:t})$ conditioned on the true path $x_{0:t}$. In practice, we do not know the path, hence one might want to know the posterior $p(m \mid z_{1:t}, u_{1:t}, c_{1:t})$ without the path in the conditioning set. This Gaussian cannot be factored in the moments parameterization, as locations of different features are correlated through the uncertainty over the robot pose. For this reason, **GraphSLAM_solve** returns the mean estimate of the posterior but only the covariance over the robot path. Luckily, we never need the full Gaussian in moments representation—which would involve a fully populated covariance matrix of massive dimensions—as all essential questions pertaining to the SLAM problem can be answered at least in approximation without knowledge of Σ.

6. Data Association in GraphSLAM

Data association in GraphSLAM is realized through correspondence variables. GraphSLAM searches for a single best correspondence vector, instead of calculating an entire distribution over correspondences. Thus, finding a correspondence vector is a search problem. However, it proves convenient to define correspondences slightly differently in GraphSLAM than before: correspondences are defined over pairs of features in the map, rather than associations of measurements to features. Specifically, we say $c(j, k) = 1$ if m_j and m_k correspond to the same physical feature in the world. Otherwise, $c(j, k) = 0$. This feature-correspondence is in fact logically equivalent to the correspondence defined in the previous section, but it simplifies the statement of the basic algorithm.

Our technique for searching the space of correspondences is greedy, just as in the EKF. Each step in the search of the best correspondence value leads to an improvement, as measured by the appropriate log-likelihood function. However, because GraphSLAM has access to all data at the same time, it is possible to devise correspondence techniques that are considerably more powerful than the incremental approach in the EKF. In particular:

1. At any point in the search, GraphSLAM can consider the correspondence of any set of features. There is no requirement to process the observed features sequentially.

2. Correspondence search can be combined with the calculation of the map. Assuming that two observed features correspond to the same physical feature in the world affects the resulting map. By incorporating such a correspondence hypothesis into the map, other correspondence hypotheses will subsequently look more or less likely.

3. Data association decisions in GraphSLAM can be undone. The goodness of a data association depends on the value of other data association decisions. What appears to be a good choice early on in the search may, at some later time in the search, turn out to be inferior. To accommodate such a situation, GraphSLAM can effectively undo a previous data association decision.

We will now describe one specific correspondence search algorithm that exploits the first two properties, but not the third.

Our data association algorithm will still be greedy, and it will sequentially search the space of possible correspondences to arrive at a plausible map. However, like all greedy algorithms, our approach is subject to local maxima; the true space of correspondences is of course exponential in the number of features in the map. Nevertheless, we will be content with a hill climbing algorithm.

6.1. The GraphSLAM Algorithm with Unknown Correspondence

The key component of our algorithm is a *likelihood test for correspondence*. Specifically, GraphSLAM correspondence is based on a simple test: what is the probability that two different features in the map, m_j and m_k, correspond to the same physical feature in the world? If this probability exceeds a threshold, we will accept this hypothesis and merge both features in the map.

The algorithm for the correspondence test is depicted in Table 9: the input to the test are two feature indexes, j and k, for which we seek to compute the probability that those two features correspond to the same feature in the physical world. To calculate this probability, our algorithm utilizes a number of quantities: the information representation of the SLAM posterior, as manifest by Ω and ξ, and the result of the procedure **GraphSLAM_solve**, which is the mean vector μ and the path covariance $\Sigma_{0:t}$.

The correspondence test then proceeds in the following way. First, it computes the marginalized posterior over the two target features. This posterior is represented by the information matrix $\Omega_{[j,k]}$ and vector $\xi_{[j,k]}$ computed in lines 2 and 3 in Table 9. This step of the computation utilizes various sub-elements of the information form Ω, ξ, the mean feature locations as specified through μ, and the path covariance $\Sigma_{0:t}$. Next, it calculates the parameters of a new Gaussian random variable, whose value is the difference between m_j and m_k. Denoting the difference variable $\Delta_{j,k} = m_j - m_k$, the information parameters $\Omega_{\Delta j,k}$, $\xi_{\Delta j,k}$ are calculated in lines 4 and 5, and the corresponding expectation for the difference is computed in line 6. Line 7 returns the probability that the difference between m_j and m_k is zero.

The correspondence test provides us with an algorithm for performing data association search in GraphSLAM. Table 10 shows such an algorithm. It initializes the correspondence variables with unique values. The four steps that follow (lines 3-7) are the same as in our GraphSLAM algorithm with known correspondence, stated in Table 5. However, this general SLAM algorithm then engages in the data association search. Specifically, for each pair of different features in the map, it calculates the probability of correspondence (line 9 in Table 10). If this probability exceeds a threshold χ, the correspondence vectors are set to the same value (line 11).

The GraphSLAM algorithm iterates the construction, reduction, and solution of the SLAM posterior (lines 12 through 14). As a result, subsequent correspondence tests factor in previous correspondence decisions though a newly constructed map. The map construction is terminated when no further features are found in its inner loop.

Clearly, the algorithm **GraphSLAM** is not particularly efficient. In particular, it tests all feature pairs for correspondence, not just nearby ones. Further, it reconstructs the map whenever a single correspondence is found; rather than processing sets of corresponding features in batch. Such modifications, however, are relatively straightforward. A good implementation of **GraphSLAM** will be more refined than our basic implementation discussed here.

6.2. Mathematical Derivation of the Correspondence Test

We essentially restrict our derivation to showing the correctness of the correspondence test in Table 9. Our first goal is to define a posterior probability distribution over a variable $\Delta_{j,k} = m_j - m_k$, the *difference* between the location of feature m_j and feature m_k. Two features m_j and m_k are equivalent if and only if their location is the same. Hence, by calculating the posterior probability of $\Delta_{j,k}$, we obtain the desired correspondence probability.

We obtain the posterior for $\Delta_{j,k}$ by first calculating the joint over m_j and m_k:

$$\begin{aligned} & p(m_j, m_k \mid z_{1:t}, u_{1:t}, c_{1:t}) \qquad (51) \\ = & \int p(m_j, m_k \mid x_{1:t}, z_{1:t}, c_{1:t})\ p(x_{1:t} \mid z_{1:t}, u_{1:t}, c_{1:t})\ dx_{1:t} \end{aligned}$$

We will denote the information form of this marginal posterior by $\xi_{[j,k]}$ and $\Omega_{[j,k]}$. Note the use of the squared brackets, which distinguish these values from the sub-matrices of the joint information form.

The distribution (51) is obtained from the joint posterior over $y_{0:t}$, by applying the marginalization lemma. Specifically, Ω and ξ represent the joint posterior over the full state vector $y_{0:t}$ in information form, and $\tau(j)$ and $\tau(k)$ denote the sets of poses at which the robot observed feature j, and feature k, respectively. GraphSLAM gives us the mean pose vector $\tilde{\mu}$. To apply the marginalization lemma (Table 7), we shall leverage the result of the algorithm **GraphSLAM_solve**. Specifically, **GraphSLAM_solve** provides us with a mean for the features m_j and m_k. We simply restate the computation here for the joint feature pair:

$$\mu_{[j,k]} = \Omega^{-1}_{jk,jk}\ (\xi_{jk} + \Omega_{jk,\tau(j,k)}\mu_{\tau(j,k)}) \qquad (52)$$

Here $\tau(j,k) = \tau(j) \cup \tau(k)$ denotes the set of poses at which the robot observed m_j or m_k.

For the joint posterior, we also need a covariance. This covariance is *not* computed in **GraphSLAM_solve**, simply because the joint covariance over multiple features requires space quadratic in the number of features. However, for pairs of features the covariance of the joint is easily recovered.

Table 9. The GraphSLAM Test for Correspondence: It Accepts as Input an Information Representation of the SLAM Posterior, Along with the Result of the GraphSLAM_solve Step. It Then Outputs the Posterior Probability That m_j Corresponds to m_k.

Algorithm GraphSLAM_correspondence_test($\Omega, \xi, \mu, \Sigma_{0:t}, j, k$):

$\Omega_{[j,k]} = \Omega_{jk,jk} - \Omega_{jk,\tau(j,k)} \; \Sigma_{\tau(j,k),\tau(j,k)} \; \Omega_{\tau(j,k),jk}$

$\xi_{[j,k]} = \Omega_{[j,k]} \; \mu_{j,k}$

$\Omega_{\Delta j,k} = \begin{pmatrix} 1 \\ -1 \end{pmatrix}^T \Omega_{[j,k]} \begin{pmatrix} 1 \\ -1 \end{pmatrix}$

$\xi_{\Delta j,k} = \begin{pmatrix} 1 \\ -1 \end{pmatrix}^T \xi_{[j,k]}$

$\mu_{\Delta j,k} = \Omega^{-1}_{\Delta j,k} \; \xi_{\Delta j,k}$

return $|2\pi \; \Omega^{-1}_{\Delta j,k}|^{-\frac{1}{2}} \; \exp\left\{-\frac{1}{2} \; \mu^T_{\Delta j,k} \; \Omega^{-1}_{\Delta j,k} \; \mu_{\Delta j,k}\right\}$

Table 10. The GraphSLAM Algorithm for the Full SLAM Problem with Unknown Correspondence. The inner loop of this algorithm can be made more efficient by selective probing feature pairs m_j, m_k, and by collecting multiple correspondences before solving the resulting collapsed set of equations.

Algorithm GraphSLAM($u_{1:t}, z_{1:t}$):

initialize all c^i_t *with a unique value*
$\mu_{0:t} =$ **GraphSLAM_initialize**$(u_{1:t})$
$\Omega, \xi =$ **GraphSLAM_linearize**$(u_{1:t}, z_{1:t}, c_{1:t}, \mu_{0:t})$
$\tilde{\Omega}, \tilde{\xi} =$ **GraphSLAM_reduce**(Ω, ξ)
$\mu, \Sigma_{0:t} =$ **GraphSLAM_solve**$(\tilde{\Omega}, \tilde{\xi}, \Omega, \xi)$
repeat
  for each pair of non-corresponding features m_j, m_k *do*
    $\pi_{j=k} =$ **GraphSLAM_correspondence_test** $(\Omega, \xi, \mu, \Sigma_{0:t}, j, k)$
    if $\pi_{j=k} > \chi$ *then*
      for all $c^i_t = k$ *set* $c^i_t = j$
      $\Omega, \xi =$ **GraphSLAM_linearize**$(u_{1:t}, z_{1:t}, c_{1:t}, \mu_{0:t})$
      $\tilde{\Omega}, \tilde{\xi} =$ **GraphSLAM_reduce**(Ω, ξ)
      $\mu, \Sigma_{0:t} =$ **GraphSLAM_solve**$(\tilde{\Omega}, \tilde{\xi}, \Omega, \xi)$
    endif
  endfor
until no more pair m_j, m_k *found with* $\pi_{j=k} < \chi$
return μ

Let $\Sigma_{\tau(j,k),\tau(j,k)}$ be the sub-matrix of the covariance $\Sigma_{0:t}$ restricted to all poses in $\tau(j,k)$. Here the covariance $\Sigma_{0:t}$ is calculated in line 2 of the algorithm **GraphSLAM_solve**. Then the marginalization lemma provides us with the marginal information matrix for the posterior over $(m_j \;\; m_k)^T$:

$$\Omega_{[j,k]} \;=\; \Omega_{jk,jk} - \Omega_{jk,\tau(j,k)}\; \Sigma_{\tau(j,k),\tau(j,k)}\; \Omega_{\tau(j,k),jk} \quad (53)$$

The information form representation for the desired posterior is now completed by the following information vector:

$$\xi_{[j,k]} \;=\; \Omega_{[j,k]}\; \mu_{[j,k]} \quad (54)$$

Hence for the joint we have:

$$\begin{aligned} & p(m_j, m_k \mid z_{1:t}, u_{1:t}, c_{1:t}) \quad (55)\\ &= \eta\; \exp\left\{-\frac{1}{2}\begin{pmatrix} m_j \\ m_k \end{pmatrix}^T \Omega_{[j,k]} \begin{pmatrix} m_j \\ m_k \end{pmatrix} + \begin{pmatrix} m_j \\ m_k \end{pmatrix}^T \xi_{[j,k]}\right\} \end{aligned}$$

These equations are identical to lines 2 and 3 in Table 9.

The useful thing about our representation is that it immediately lets us define the desired correspondence probability. For that, let us consider the random variable:

$$\begin{aligned} \Delta_{j,k} &= m_j - m_k \quad (56)\\ &= \begin{pmatrix} 1 \\ -1 \end{pmatrix}^T \begin{pmatrix} m_j \\ m_k \end{pmatrix} \\ &= \begin{pmatrix} m_j \\ m_k \end{pmatrix}^T \begin{pmatrix} 1 \\ -1 \end{pmatrix} \end{aligned}$$

Plugging this into the definition of a Gaussian in information representation, we obtain:

$$\begin{aligned} & p(\Delta_{j,k} \mid z_{1:t}, u_{1:t}, c_{1:t}) \quad (57)\\ &= \eta \exp\left\{-\frac{1}{2}\Delta_{j,k}^T \underbrace{\begin{pmatrix} 1 \\ -1 \end{pmatrix}^T \Omega_{[j,k]} \begin{pmatrix} 1 \\ -1 \end{pmatrix}}_{=:\,\Omega_{\Delta j,k}} \Delta_{j,k} + \Delta_{j,k}^T \underbrace{\begin{pmatrix} 1 \\ -1 \end{pmatrix}^T \xi_{[j,k]}}_{=:\,\xi_{\Delta j,k}}\right\} \\ &= \eta\; \exp\left\{-\frac{1}{2}\; \Delta_{j,k}^T\; \Omega_{\Delta j,k} \;+ \Delta_{j,k}^T\; \xi_{\Delta j,k}\right\}^T \end{aligned}$$

which is Gaussian with the information matrix $\Omega_{\Delta j,k}$ and information vector $\xi_{\Delta j,k}$ as defined above. To calculate the probability that this Gaussian assumes the value of $\Delta_{j,k} = 0$, it is useful to rewrite this Gaussian in moments parameterization:

$$\begin{aligned} & p(\Delta_{j,k} \mid z_{1:t}, u_{1:t}, c_{1:t}) \quad (58)\\ &= |2\pi\; \Omega_{\Delta j,k}^{-1}|^{-\frac{1}{2}} \\ &\quad \exp\left\{-\frac{1}{2}\; (\Delta_{j,k} - \mu_{\Delta j,k})^T\; \Omega_{\Delta j,k}^{-1}\; (\Delta_{j,k} - \mu_{\Delta j,k})\right\} \end{aligned}$$

where the mean is given by the obvious expression:

$$\mu_{\Delta j,k} \;=\; \Omega_{\Delta j,k}^{-1}\; \xi_{\Delta j,k} \quad (59)$$

These steps are found in lines 4 through 6 in Table 9.

The desired probability for $\Delta_{j,k} = 0$ is the result of plugging 0 into this distribution, and reading off the resulting probability:

$$\begin{aligned} p(\Delta_{j,k} = 0 \mid z_{1:t}, u_{1:t}, c_{1:t}) &= |2\pi\; \Omega_{\Delta j,k}^{-1}|^{-\frac{1}{2}} \quad (60)\\ &\exp\left\{-\frac{1}{2}\; \mu_{\Delta j,k}^T\; \Omega_{\Delta j,k}^{-1}\; \mu_{\Delta j,k}\right\} \end{aligned}$$

This expression is the probability that two features in the map, m_j and m_k, correspond to the same features in the map. This calculation is implemented in line 7 in Table 9.

7. Results

We conducted a number of experiments, all with the robot shown in Figure 4. In particular, we mapped a number of urban sites, including NASA's Search and Rescue Facility DART and a large fraction of Stanford's main campus; snapshots of these experiments will be discussed below.

Our experiments either involved the collection of a single large dataset, or a number of datasets. The latter became necessary since for the environments of the size studied here, the robot possesses insufficient battery capacity to collect all data within a single run. In most experiments, the robot is controlled manually. This is necessary because the urban environments are usually populated with moving objects, such as cars, which would otherwise run the danger of colliding with our robot. We have, on several occasions, used our navigation package Carmen (Montemerlo, Roy, and Thrun 2003) to drive the robot autonomously, validating the terrain analysis techniques discussed above.

Our research has led to a number of results. A primary finding is that with our representation, maps with more than 10^8 variables can be computed quickly, even under multiple loop-closure constraints. The time for thinning the network into its skeleton tends to take linear time in the number of robot poses, which is the same order as the time required for data collection. We find that scan matching is easily achieved in real-time, as the robot moves, using a portable laptop computer. This is a long-known result for horizontally mounted

Fig. 4. The Segbot, a robot based on the Segway RMP platform and developed through the DARPA MARS program.

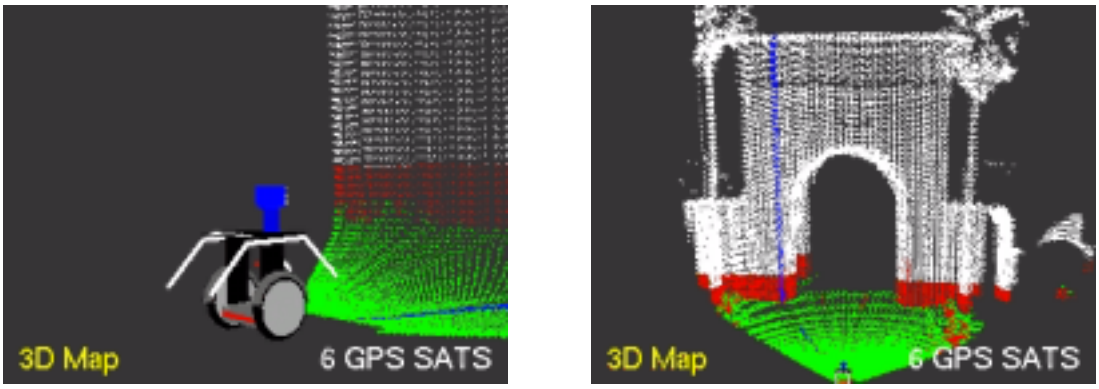


Fig. 5. Data acquisition through a two-directional scanning laser (the blue stripe indicates a vertical scan). The coloring indicates the result of terrain analysis: The ground surface is colored in green, obstacles are red, and structure above the robot's reach are shown in white.

laser range finders, but it is reassuring that the same applies to the more difficult scan matching problem involving a vertically panning scanner. More importantly, the relaxation of the pose potentials takes in the order of 30 seconds even for the largest data set used in our research, of an area 600 m by 800 m in size, and with a dozen cycles. This suggests the appropriateness of our representation an algorithms for large-scale urban mapping.

The second result pertains to the utility of GPS data for indoor maps. GPS measurements are easily incorporated into GraphSLAM; they form yet another arc in the graph of constraints. We find that indoor maps become more accurate when some of the data is collected outdoors, where GPS measurements are available. Below, we will discuss an experimental snapshot that documents this result.

Experimental snapshots can be found in Figures 6 through 8. Figures 7 and 8 show some of the maps acquired by our system. All maps are substantially larger than previously software could handle, all are constructed with some GPS information. The map shown on the top in Figure 7 corresponds to Stanford's main campus; the one on the bottom is an indoor-outdoor map of the building that houses the computer science department.

The key result of improved indoor maps through combining indoor and outdoor mapping is illustrated in Figure 6. Here we show 2-D slices of the 3-D map in Figure 7 using SLAM under two different conditions: In the map on the top, the indoor map is constructed independently of the outdoor map, whereas the bottom map is constructed jointly. As explained, the joint construction lets GPS information affect the building interior through the sequence of potentials liking the outdoor to the indoor. As this figure suggests, the joint indoor-outdoor map is significantly more accurate; in fact, the building possesses a right angle at its center, which is well approximated.

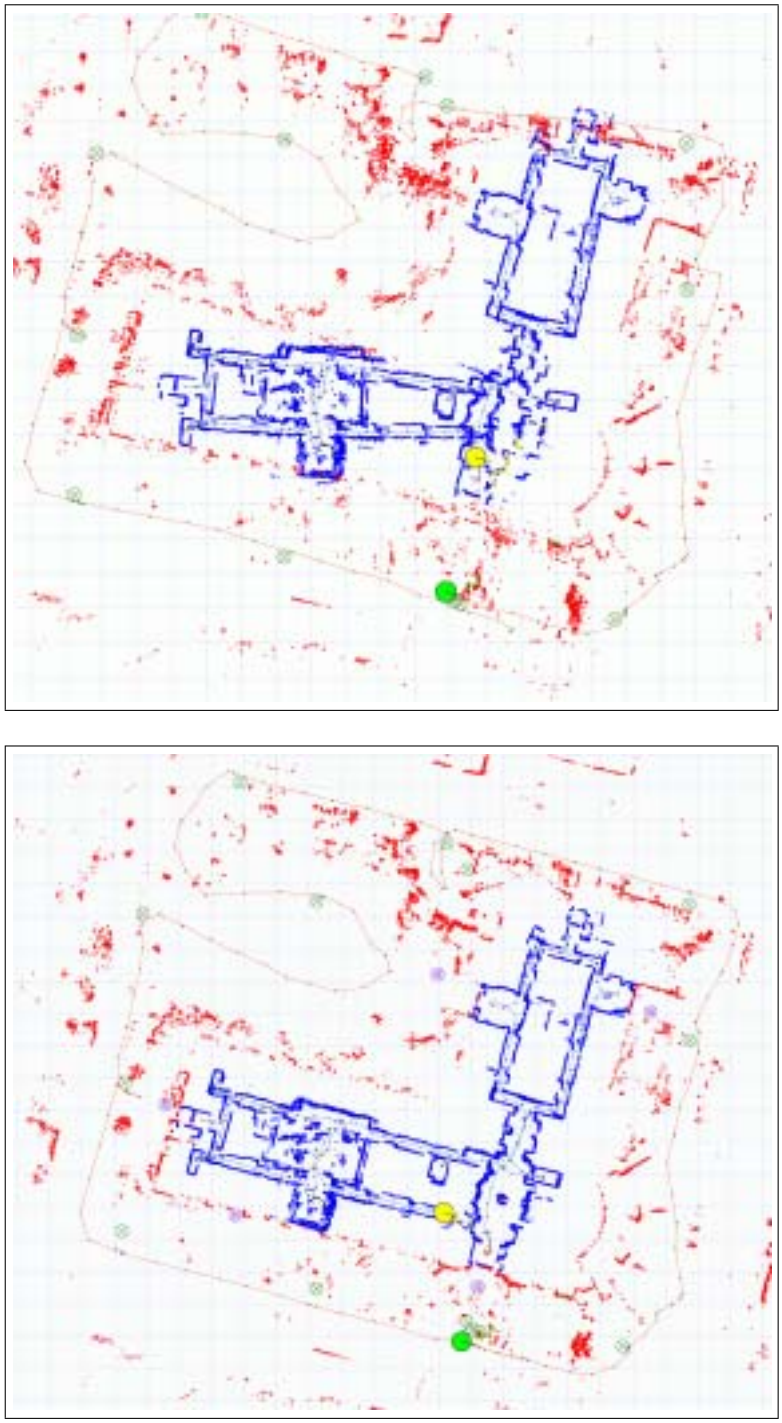


Fig. 6. Indoor mapping. Top: just based on the IMU and SLAM. Bottom: factoring in GPS data acquired outdoors. This experiment highlights the utility of our hybrid SLAM algorithm that factors in GPS measurements as available.

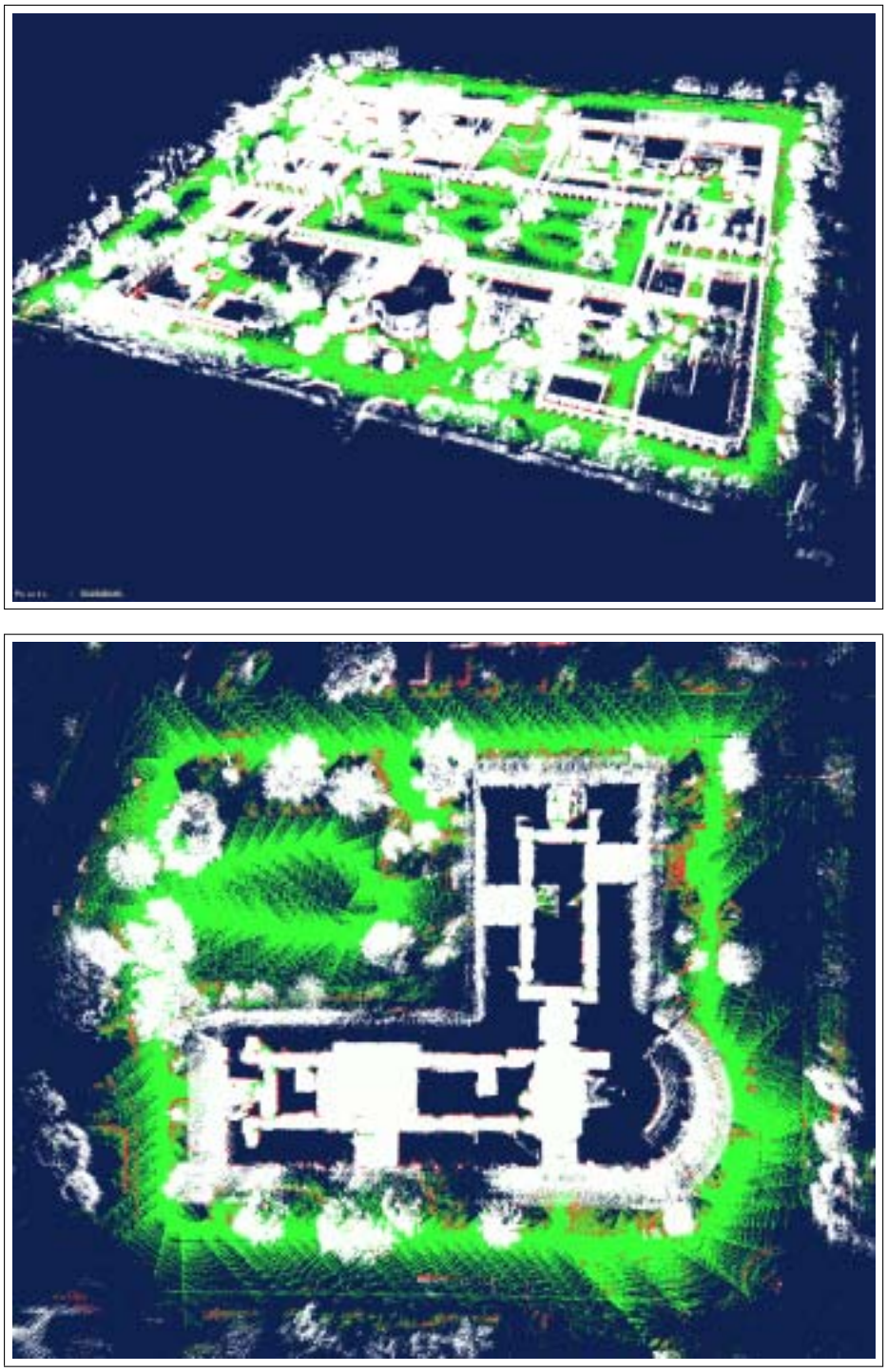


Fig. 7. Top: A map of Stanford University's main campus, whose diameter is approximately 600 meters. Bottom: 3-D map of the Gates Computer Science building and the surrounding terrain.

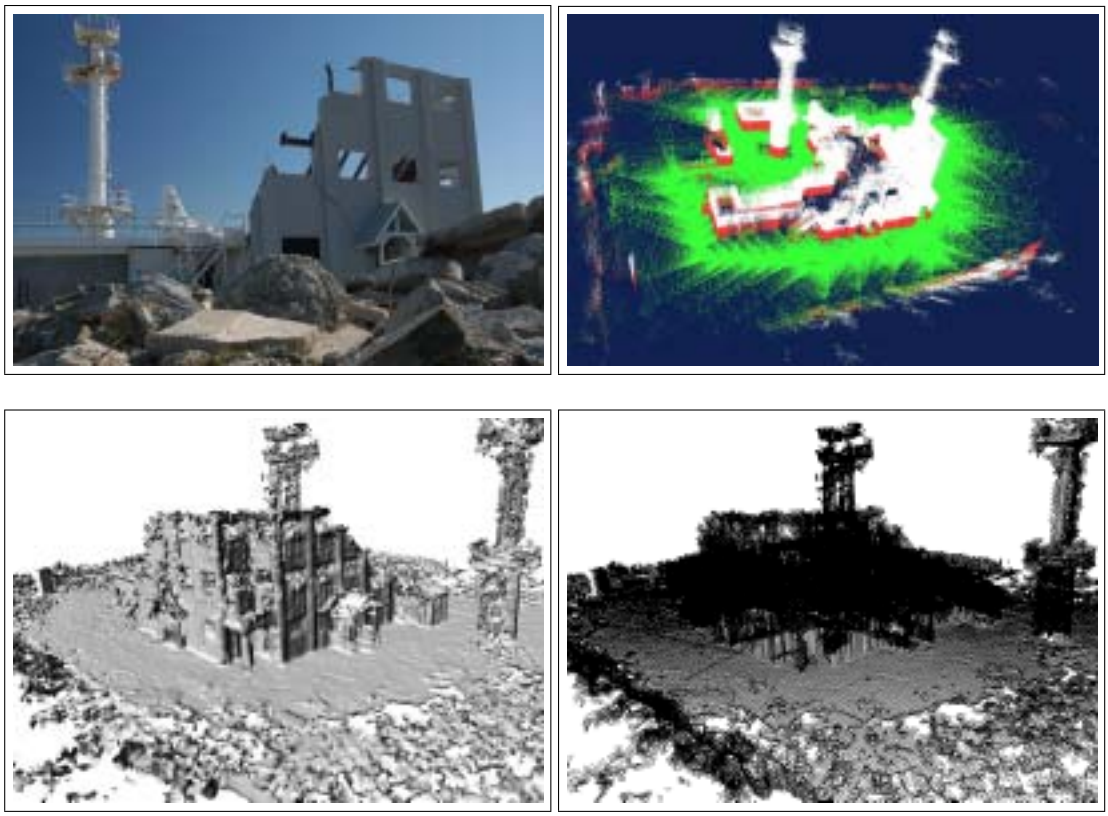


Fig. 8. Visualization of the NASA Ames ARC Disaster Assistance and Rescue Team training site in Moffett Field, CA. This site consist of a partially collapsed building with two large observation platforms.

8. Conclusion

We presented the GraphSLAM algorithm, which solves a specific version of the SLAM problem, called the offline problem (or full SLAM problem). The offline problem is characterized by a feasibility to accumulate all data during mapping, and resolve this data into a map after the robot's operation is complete. GraphSLAM achieves the latter by mapping the data into a sparse graph of constraints, which are then mapped into an information form representation using linearization through Taylor expansion. The information form is then reduced by applying exact transformations, which remove the map variables from the optimization problem. The resulting optimization problem is solved via a standard optimization technique, such as conjugate gradient. GraphSLAM recovers the map from the pose estimate, through a sequence of decoupled small-scale optimization problems (one per feature). Iteration of the linearization and optimization technique yields accurate maps in environments with 10^8 features or more.

The GraphSLAM algorithm follows a rich tradition of previously published offline SLAM algorithms, which are all based on the insight that the full SLAM problem corresponds to a sparse spring-mass system. The key innovation in this paper is the reduction step, through which the problem of inference in this graphical model becomes manageable. This step is essential in achieving scalability in offline SLAM.

Experimental results in large-scale urban environments show that the GraphSLAM approach indeed leads to viable maps. Our experiments show that it is relatively straightforward to include other information sources—such as GPS—into the SLAM problem, by defining appropriate graphical constraints. For example, we were able to show that through the graphical model, GPS data acquired outside a building structure could be propagated into the building interior, thereby improving the accuracy of an interior map.

GraphSLAM is characterized by a number of limitations. One arises from the assumption of independent Gaussian noise. Clearly, real-world noise is not Gaussian and, more importantly, it is not independent. We find in practice that this problem can be alleviated by artificially increasing the covariance of the noise variables, which reduces the information available for SLAM. However, such methods are somewhat ad-hoc; see Guivant and Masson (2005) for further treatment of non-Gaussian noise.

GraphSLAM is also limited in its reliance on a good initial estimate of the map, computed in Table 1. As the total number of time steps increases, the accuracy of the odometry-based initial guess will degrade, leading to an increased number of data association errors. GraphSLAM, as formulated in this paper, will eventually diverge because of this initial pose estimation step. However, the data set may often be broken into pieces, for which SLAM can be performed individually before pasting together the total set of constraints. Such a hierarchical computation is subject of future research.

Another limitation pertains to the matrix inversion in Table 4. This inversion can be painfully slow; and optimization methods such as conjugate gradient (as brought to bear in our experiments, can be much more efficient.

More broadly, there remain a number of open questions that warrant future research. Chief among them is the development of SLAM techniques that can handle basic building elements, such as walls, windows, roofs, and so on. GraphSLAM makes a static world assumption, and more research is needed to understand SLAM in dynamic environments (see Hähnel, Schulz, and Burgard 2003; Wang, Thorpe, and Thrun 2003 for notable exceptions). Finally, bridging the gap between online and offline SLAM algorithm is a worthwhile goal of future research.

Appendix: Derivations

The derivations in this section are standard textbook material.

Derivation of the Inversion Lemma

Define $\Psi = (Q^{-1} + P^T\ R^{-1}\ P)^{-1}$. It suffices to show that

$$(R^{-1} - R^{-1}\ P\ \Psi\ P^T\ R^{-1})\ (R + P\ Q\ P^T) \quad = \quad I$$

This is shown through a series of transformations:

$$\begin{aligned}
&= \underbrace{R^{-1}\ R}_{=I} + R^{-1}\ P\ Q\ P^T - R^{-1}\ P\ \Psi\ P^T\ \underbrace{R^{-1}\ R}_{=I} \\
&\quad - R^{-1}\ P\ \Psi\ P^T\ R^{-1}\ P\ Q\ P^T \\
&= I + R^{-1}\ P\ Q\ P^T - R^{-1}\ P\ \Psi\ P^T \\
&\quad - R^{-1}\ P\ \Psi\ P^T\ R^{-1}\ P\ Q\ P^T \\
&= I + R^{-1}\ P\ [Q\ P^T \\
&\quad -\Psi\ P^T - \Psi\ P^T\ R^{-1}\ P\ Q\ P^T] \\
&= I + R^{-1}\ P\ [Q\ P^T - \Psi\ \underbrace{Q^{-1}\ Q}_{=I}\ P^T \\
&\quad -\Psi\ P^T\ R^{-1}\ P\ Q\ P^T] \\
&= I + R^{-1}\ P\ [Q\ P^T - \underbrace{\Psi\ \Psi^{-1}}_{=I}\ Q\ P^T] \\
&= I + R^{-1}\ P\ [\underbrace{Q\ P^T - Q\ P^T}_{=0}] \quad = \quad I
\end{aligned}$$

Marginals of a Multivariate Gaussian

The marginal for a Gaussian in its moments parameterization

$$\Sigma \quad = \quad \begin{pmatrix} \Sigma_{xx} & \Sigma_{xy} \\ \Sigma_{yx} & \Sigma_{yy} \end{pmatrix} \quad \text{and} \quad \mu \quad = \quad \begin{pmatrix} \mu_x \\ \mu_y \end{pmatrix}$$

is $\mathcal{N}(\mu_x, \Sigma_{xx})$. By definition, the information matrix of this Gaussian is therefore Σ_{xx}^{-1}, and the information vector is $\Sigma_{xx}^{-1}\ \mu_x$. We show $\Sigma_{xx}^{-1} = \bar{\Omega}_{xx}$ via the Inversion Lemma from Table 6; this derivation makes the assumption that none of the participating matrices is singular. Let $P = (0\ \ 1)^T$, and let $[\infty]$ be a matrix of the same size as Ω_{yy} but whose entries are all infinite (and with $[\infty]^{-1} = 0$. This gives us

$$(\Omega + P[\infty]P^T)^{-1} = \begin{pmatrix} \Omega_{xx} & \Omega_{xy} \\ \Omega_{yx} & [\infty] \end{pmatrix}^{-1} \overset{(*)}{=} \begin{pmatrix} \Sigma_{xx}^{-1} & 0 \\ 0 & 0 \end{pmatrix}$$

The same expression can also be expanded by the inversion lemma into:

$$\begin{aligned}
&(\Omega + P[\infty]P^T)^{-1} \\
&\quad = \quad \Omega - \Omega\ P([\infty]^{-1} + P^T\ \Omega\ P)^{-1}\ P^T\ \Omega \\
&\quad = \quad \Omega - \Omega\ P(0 + P^T\ \Omega\ P)^{-1}\ P^T\ \Omega \\
&\quad = \quad \Omega - \Omega\ P(\Omega_{yy})^{-1}\ P^T\ \Omega \\
&\quad = \quad \begin{pmatrix} \Omega_{xx} & \Omega_{xy} \\ \Omega_{yx} & \Omega_{yy} \end{pmatrix} - \begin{pmatrix} \Omega_{xx} & \Omega_{xy} \\ \Omega_{yx} & \Omega_{yy} \end{pmatrix} \begin{pmatrix} 0 & 0 \\ 0 & \Omega_{yy}^{-1} \end{pmatrix} \\
&\qquad \begin{pmatrix} \Omega_{xx} & \Omega_{xy} \\ \Omega_{yx} & \Omega_{yy} \end{pmatrix} \\
&\quad \overset{(*)}{=} \quad \begin{pmatrix} \Omega_{xx} & \Omega_{xy} \\ \Omega_{yx} & \Omega_{yy} \end{pmatrix} - \begin{pmatrix} 0 & \Omega_{xy}\ \Omega_{yy}^{-1} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \Omega_{xx} & \Omega_{xy} \\ \Omega_{yx} & \Omega_{yy} \end{pmatrix} \\
&\quad = \quad \begin{pmatrix} \Omega_{xx} & \Omega_{xy} \\ \Omega_{yx} & \Omega_{yy} \end{pmatrix} - \begin{pmatrix} \Omega_{xy}\ \Omega_{yy}^{-1}\ \Omega_{yx} & \Omega_{xy} \\ \Omega_{yx} & \Omega_{yy} \end{pmatrix} \\
&\quad = \quad \begin{pmatrix} \bar{\Omega}_{xx} & 0 \\ 0 & 0 \end{pmatrix}
\end{aligned}$$

The remaining statement, $\Sigma_{xx}^{-1}\ \mu_x = \bar{\xi}_x$, is obtained analogously, exploiting the fact that $\mu = \Omega^{-1}\xi$ and the equality of the two expressions marked "$(*)$" above:

$$\begin{aligned}
\begin{pmatrix} \Sigma_{xx}^{-1}\ \mu_x \\ 0 \end{pmatrix} \quad &= \quad \begin{pmatrix} \Sigma_{xx}^{-1} & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \mu_x \\ \mu_y \end{pmatrix} \\
&= \quad \begin{pmatrix} \Sigma_{xx}^{-1} & 0 \\ 0 & 0 \end{pmatrix} \Omega^{-1} \begin{pmatrix} \xi_x \\ \xi_y \end{pmatrix} \\
&\overset{(*)}{=} \quad \left[\Omega - \begin{pmatrix} 0 & \Omega_{xy}\ \Omega_{yy}^{-1} \\ 0 & 1 \end{pmatrix} \Omega\right] \Omega^{-1} \begin{pmatrix} \xi_x \\ \xi_y \end{pmatrix} \\
&= \quad \begin{pmatrix} \xi_x \\ \xi_y \end{pmatrix} - \begin{pmatrix} 0 & \Omega_{xy}\ \Omega_{yy}^{-1} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \xi_x \\ \xi_y \end{pmatrix} \\
&= \quad \begin{pmatrix} \bar{\xi}_x \\ 0 \end{pmatrix}
\end{aligned}$$

Conditionals of a multivariate Gaussian

Proof. The result follows trivially from the definition of a Gaussian in information form:

$$
\begin{aligned}
& p(x \mid y) \\
&= \eta \ \exp\left\{-\frac{1}{2}\begin{pmatrix} x \\ y \end{pmatrix}^T \begin{pmatrix} \Omega_{xx} & \Omega_{xy} \\ \Omega_{yx} & \Omega_{yy} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \right. \\
&\quad \left. + \begin{pmatrix} x \\ y \end{pmatrix}^T \begin{pmatrix} \xi_x \\ \xi_y \end{pmatrix} \right\} \\
&= \eta \ \exp\left\{-\frac{1}{2} x^T \Omega_{xx} x + x^T \Omega_{xy} y - \frac{1}{2} \right. \\
&\quad \left. + y^T \Omega_{yy} y + x^T \xi_x + y^T \xi_y \right\} \\
&= \eta \ \exp\{-\frac{1}{2} x^T \Omega_{xx} x \\
&\quad + x^T (\Omega_{xy} y + \xi_x) \underbrace{-\frac{1}{2} + y^T \Omega_{yy} y + y^T \xi_y}_{\text{const.}} \} \\
&= \eta \ \exp\{-\frac{1}{2} x^T \Omega_{xx} x + x^T (\Omega_{xy} y + \xi_x)\}
\end{aligned}
$$

References

Allen, P. and Stamos, I. 2000. Integration of range and image sensing for photorealistic 3D modeling. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1435–1440.

Araneda, A. 2003. Statistical inference in mapping and localization for a mobile robot. In *Bayesian Statistics 7*. (eds Bernardo, J. M., Bayarri, M., Berger, J., Dawid, A. P., Heckerman, D., Smith, A., and West, M). Oxford University Press, Oxford, UK.

Bailey, T. 2002. *Mobile Robot Localisation and Mapping in Extensive Outdoor Environments*. PhD thesis, University of Sydney, Sydney, NSW, Australia.

Bajcsy, R., Kamberova, G., and Nocera, L. 2000. 3D reconstruction of environments for virtual reconstruction. In *Proc. of the 4th IEEE Workshop on Applications of Computer Vision*.

Baker, C., Morris, A., Ferguson, D., Thayer, S., Whittaker, C., Omohundro, Z., Reverte, C., Whittaker, W., Hähnel, D., and Thrun, S. 2004. A campaign in autonomous mine mapping. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

Bosse, M., Newman, P., Leonard, J., Soika, M., Feiten, W., and Teller, S. 2004. Simultaneous localization and map building in large-scale cyclic environments using the atlas framework. *International Journal of Robotics Research* 23(12):1113–1139.

Bosse, M., Newman, P., Soika, M., Feiten, W., Leonard, J., and Teller, S. 2003. An atlas framework for scalable mapping. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

Bulata, H. and Devy, M. 1996. Incremental construction of a landmark-based and topological model of indoor environments by a mobile robot. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Minneapolis, USA.

Cheeseman, P. and Smith, P. 1986. On the representation and estimation of spatial uncertainty. *International Journal of Robotics* 5:56–68.

Cowell, R., Dawid, A., Lauritzen, S., and Spiegelhalter, D. 1999. *Probabilistic Networks and Expert Systems*. Springer Verlag, Berlin, New York.

Csorba, M. 1997. *Simultaneous Localisation and Map Building*. PhD thesis, University of Oxford.

Dissanayake, G., Newman, P., Clark, S., Durrant-Whyte, H., and Csorba, M. 2001. A solution to the simultaneous localisation and map building (SLAM) problem. *IEEE Transactions of Robotics and Automation* 17(3):229–241.

Duckett, T., Marsland, S., and Shapiro, J. 2000. Learning globally consistent maps by relaxation. In *Proceedings of the IEEE International Conference on Robotics and Automation, San Francisco*, pp. 3841–3846.

Duckett, T., Marsland, S., and Shapiro, J. 2002. Fast, on-line learning of globally consistent maps. *Autonomous Robots* 12(3):287–300.

Durrant-Whyte, H. 1988. Uncertain geometry in robotics. *IEEE Transactions on Robotics and Automation* 4(1):23–31.

El-Hakim, S., Boulanger, P., Blais, F., and Beraldin, J.-A. 1997. Sensor based creation of indoor virtual environment models. In *Proc. of the 4th Internetional Conference on Virtual Systems and Multimedia (VSMM)*, Geneva, Switzerland.

Elfes, A. 1987. Sonar-based real-world mapping and navigation. *IEEE Journal of Robotics and Automation* RA-3(3):249–265.

Folkesson, J. and Christensen, H. I. 2004a. Graphical SLAM: A self-correcting map. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, New Orleans, USA.

Folkesson, J. and Christensen, H. I. 2004b. Robust SLAM. In *Proceedings of the International Symposium on Autonomous Vehicles*, Lisboa, PT.

Frese, U. 2004. *An $O(logn)$ Algorithm for Simultaneous Localization and Mapping of Mobile Robots in Indoor Environments*. PhD thesis, University of Erlangen-Nürnberg, Germany.

Frese, U. and Hirzinger, G. 2001. Simultaneous localization and mapping—a discussion. In *Proceedings of the IJCAI Workshop on Reasoning with Uncertainty in Robotics*, pp. 17–26, Seattle, WA.

Frese, U., Larsson, P., and Duckett, T. 2005. A Multigrid Algorithm for Simultaneous Localization and Mapping. In *IEEE Transactions on Robotics*, 21(2):1-12.

Golfarelli, M., Maio, D., and Rizzi, S. 1998. Elastic correction of dead-reckoning errors in map building. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 905–911, Victoria, Canada.

Guivant, J. and Masson, F. 2005. Using absolute non-gaussian non-white observations in Gaussian SLAM. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Barcelona, Spain.

Guivant, J. and Nebot, E. 2001. Optimization of the simultaneous localization and map building algorithm for real time implementation. *IEEE Transactions of Robotics and Automation* 17(3):242–257.

Gutmann, J.-S. and Konolige, K. 2000. Incremental mapping of large cyclic environments. In *Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA)*.

Gutmann, J.-S. and Nebel, B. 1997. Navigation mobiler roboter mit laserscans. In *Autonome Mobile Systeme*. Springer Verlag, Berlin. In German.

Hähnel, D., Burgard, W., Wegbreit, B., and Thrun, S. 2003a. Towards lazy data association in SLAM. In *Proceedings of the 11th International Symposium of Robotics Research (ISRR'03)*, Sienna, Italy. Springer.

Hähnel, D., Schulz, D., and Burgard, W. 2003b. Mobile robot mapping in populated environments. *Autonomous Robots* 17(7):579–598.

Iocchi, L., Konolige, K., and Bajracharya, M. 2000. Visually realistic mapping of a planar environment with stereo. In *Proceesings of the 2000 International Symposium on Experimental Robotics*, Waikiki, Hawaii.

Julier, S. and Uhlmann, J. K. 2000. Building a million beacon map. In *Proceedings of the SPIE Sensor Fusion and Decentralized Control in Robotic Systems IV*, Vol. #4571.

Konecny, G. 2002. *Geoinformation: Remote Sensing, Photogrammetry and Geographical Information Systems*. Taylor & Francis.

Konolige, K. 2004. Large-scale map-making. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, pp. 457–463, San Jose, CA. AAAI.

Kuipers, B., Modayil, J., Beeson, P., MacMahon, M., and Savelli, F. 2004. Local metrical and global topological maps in the hybrid spatial semantic hierarchy. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, New Orleans.

Lawler, E. and Wood, D. 1966. Branch-and-bound methods: A survey. *Operations Research* 14:699–719.

Leonard, J. J. and Durrant-Whyte, H. F. 1991. Mobile robot localization by tracking geometric beacons. *IEEE Transactions on Robotics and Automation* 7(3):376–382.

Leonard, J. and Newman, P. 2003. Consistent, convergent, and constant-time SLAM. In *Proceedings of the IJCAI Workshop on Reasoning with Uncertainty in Robot Navigation*, Acapulco, Mexico.

Leonard, J., Tardós, J., Thrun, S., and Choset, H. (eds) 2002. *Workshop Notes of the ICRA Workshop on Concurrent Mapping and Localization for Autonomous Mobile Robots (W4)*. ICRA Conference, Washington, DC.

Levoy, M. 1999. The digital michelangelo project. In *Proc. of the Second International Conference on 3D Imaging and Modeling*.

Liu, Y. and Thrun, S. 2003. Results for outdoor-SLAM using sparse extended information filters. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

Lu, F. and Milios, E. 1997. Globally consistent range scan alignment for environment mapping. *Autonomous Robots* 4:333–349.

Montemerlo, M., Roy, N., and Thrun, S. 2003. Perspectives on standardization in mobile robot programming: The Carnegie Mellon navigation (CARMEN) toolkit. In *Proceedings of the Conference on Intelligent Robots and Systems (IROS)*. Software package for download at www.cs.cmu.edu/∼carmen.

Montemerlo, M. and Thrun, S. 2004. Large-scale robotic 3-d mapping of urban structures. In *Proceedings of the International Symposium on Experimental Robotics (ISER)*, Singapore. Springer Tracts in Advanced Robotics (STAR).

Montemerlo, M., Thrun, S., Koller, D., and Wegbreit, B. 2002. FastSLAM: A factored solution to the simultaneous localization and mapping problem. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, Edmonton, Canada. AAAI.

Moravec, H. P. 1988. Sensor fusion in certainty grids for mobile robots. *AI Magazine* 9(2):61–74.

Moutarlier, P. and Chatila, R. 1989a. An experimental system for incremental environment modeling by an autonomous mobile robot. In *1st International Symposium on Experimental Robotics*, Montreal.

Moutarlier, P. and Chatila, R. 1989b. Stochastic multisensory data fusion for mobile robot location and environment modeling. In *5th Int. Symposium on Robotics Research*, Tokyo.

Narendra, P. and Fukunaga, K. 1977. A branch and bound algorithm for feature subset selection. *IEEE Transactions on Computers* 26(9):914–922.

Newman, P. 2000. *On the Structure and Solution of the Simultaneous Localisation and Map Building Problem*. PhD thesis, Australian Centre for Field Robotics, University of Sydney, Sydney, Australia.

Newman, P. M. and Durrant-Whyte, H. F. 2001. A new solution to the simultaneous and map building (SLAM) problem. In *Proceedings of SPIE*.

Newman, P. and Rikoski, J. L. R. 2003. Towards constant-time slam on an autonomous underwater vehicle using synthetic aperture sonar. In *Proceedings of the International Symposium of Robotics Research*, Sienna, Italy.

Paskin, M. 2003. Thin junction tree filters for simultaneous

localization and mapping. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)*, Acapulco, Mexico. IJCAI.

Pearl, J. 1988. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann Publishers, San Mateo, CA.

Pollefeys, M., Koch, R., and Gool, L. V. 1998. Self-calibration and metric reconstruction in spite of varying and unknown internal camera parameters. In *Proceedings of the International Conference on Computer Vision*, pp. 90–95, Bombay, India. IEEE.

Rusinkiewicz, S. and Levoy, M. 2001. Efficient variants of the ICP algorithm. In *Proc. Third International Conference on 3D Digital Imaging and Modeling (3DIM)*, Quebec City, Canada. IEEEComputer Society.

Smith, R. and Cheeseman, P. 1986. On the representation and estimation of spatial uncertainty. *International Journal of Robotics Research* 5(4):56–68.

Smith, R., Self, M., and Cheeseman, P. 1990. Estimating uncertain spatial relationships in robotics. In *Autonomous Robot Vehicles* (eds Cox, I. and Wilfong, G.) pp. 167–193. Springer-Verlag.

Soatto, S. and Brockett, R. 1998. Optimal structure from motion: Local ambiguities and global estimates. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 282–288.

Tardós, J., Neira, J., Newman, P., and Leonard, J. 2002. Robust mapping and localization in indoor environments using sonar data. *International Journal of Robotics Research* 21(4):311–330.

Teller, S., Antone, M., Bodnar, Z., Bosse, M., Coorg, S., Jethwa, M., and Master, N. 2001. Calibrated, registered images of an extended urban area. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*.

Thrun, S., Koller, D., Ghahramani, Z., Durrant-Whyte, H., and Ng, A. 2002. Simultaneous mapping and localization with sparse extended information filters. In *Proceedings of the Fifth International Workshop on Algorithmic Foundations of Robotics* (eds Boissonnat, J.-D., Burdick, J., Goldberg, K., and Hutchinson, S.), Nice, France.

Thrun, S. and Liu, Y. 2003. Multi-robot SLAM with sparse extended information filers. In *Proceedings of the 11th International Symposium of Robotics Research (ISRR'03)*, Sienna, Italy. Springer.

Tomasi, C. and Kanade, T. 1992. Shape and motion from image streams under orthography: A factorization method. *International Journal of Computer Vision* 9(2):137–154.

Uhlmann, J., Lanzagorta, M., and Julier, S. 1999. The NASA mars rover: A testbed for evaluating applications of covariance intersection. In *Proceedings of the SPIE 13th Annual Symposium in Aerospace/Defence Sensing, Simulation and Controls*.

Wang, C.-C., Thorpe, C., and Thrun, S. 2003. Online simultaneous localization and mapping with detection and tracking of moving objects: Theory and results from a ground vehicle in crowded urban areas. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

Williams, S. B. 2001. *Efficient Solutions to Autonomous Mapping and Navigation Problems*. PhD thesis, ACFR, University of Sydney, Sydney, Australia.

Williams, S., Dissanayake, G., and Durrant-Whyte, H. 2001. Towards terrain-aided navigation for underwater robotics. *Advanced Robotics* 15(5).